```
                                           ;7533   ;HERE FOR CLRQ=CLRD
                                           ;7534
                                           ;7535   04D:     ;-----------------------------;
U 004D, 0018,0C38,1980,F980,0000,004F      ;7536   CLRQ:    RC[T0]_K[ZERO]                  ;CLRQ/CLRD - MAKE TWO LONGWORDS OF ZERO
                                           ;7537
                                           ;7538   ;HERE FOR CLRx, x=B, W, L (CLRF=CLRL)
                                           ;7539
                                           ;7540   04F:     ;-----------------------------;
                                           ;7541   CLR:     ALU_K[ZERO],D_0,                ;SETUP ZERO TO STORE
U 004F, FF18,003B,19F0,F847,0000,0200      ;7542            B.FORK                          ;GO STORE IT
                                           ;7543
                                           ;7544   ;HERE FOR MOVPSL
                                           ;7545
                                           ;7546   04E:     ;-----------------------------;
U 004E, 0000,003C,3DF0,2C00,0000,00B9      ;7547   MOVPSL:  Q_ID[PSL]                       ;READ PSL OVER ID BUS
                                           ;7548
                                           ;7549            ;-----------------------------;
                                           ;7550   WRQ.DST:D_Q,                             ;MOVE TO D FOR STORAGE
U 00B9, FC00,003F,01F0,F847,0000,0300      ;7551            WRITE.DEST,J/WRD                ;STORE IT, DO NOT CHANGE CC
```

```
                                               ;HERE ARE EXECUTION STATES FOR ESCAPES AND RESERVED INSTRUCTIONS

                                               081:    ;---------------------------------;
U 0081, 0018,0038,6580,F980,0000,08FC                  RC[T0]_K[.10],J/EXCPT             ;ESCE - FAULT THROUGH 10

                                               083:    ;---------------------------------;
U 0083, 0018,0038,6580,F980,0000,08FC                  RC[T0]_K[.10],J/EXCPT             ;ESCF - FAULT THROUGH 10

                                               085:    ;---------------------------------;
U 0085, 0018,0038,6580,F980,0000,08FC                  RC[T0]_K[.10],J/EXCPT             ;ESCD - FAULT THROUGH 10

                                               ; ************************************************
                                               ; * Patch no. 070, PCS 0085 trapped to WCS 1180 *
                                               ; ************************************************

                                               087:    ;---------------------------------;
U 0087, 0018,0038,6580,F980,0000,08FC                  RC[T0]_K[.10],J/EXCPT             ;RESERVED OPCODE - FAULT THROUGH 10

                                               086:    ;---------------------------------;
U 0086, 0018,0038,2180,F980,0000,08FC                  RC[T0]_K[.14],J/EXCPT             ;ESCC - FAULT THROUGH 14

                                               ;HERE FOR RSB

                                               08A:    ;---------------------------------;
U 008A, 0000,003C,0180,FA70,0200,00C6          RSB:    VA_R[SP]                          ;ADDRESS OF TOP OF STACK

                                                       ;---------------------------------;
                                                       R[SP]_LA+K[.4].RLOG,              ;UPDATE STACK POINTER
                                                       D[LONG]_CACHE,                    ;GET RETURN ADDR,
U 00C6, 0018,0018,1180,42F0,0000,028E                  J/JMP                             ;JUMP TO IT

                                               ;HERE FOR BREAKPOINT

                                               08C:    ;---------------------------------;BREAKPOINT
                                               BPT:    RC[T0]_K[.B0].RIGHT2,             ;VECTOR ADDRESS IS 2C
U 008C, 0098,0038,9580,F980,0000,08FC                  J/EXCPT                           ;TAKE THE FAULT

                                               ;HERE IS [illegible] OPERATION

                                               08E:    ;-[illegible]-----------------------------;
                                               NOP:    [illegible]B.OPC,                 ;DO NOTHING MUCH
U 008E, C000,003C,0180,F804,4000,0062                  PC[illegible]1,J/IRD

                                               08F:    ;-[illegible]------------------------;
                                               HALT:   D_K[illegible]                    ;DO EVEN LESS
U 008F, 0818,1C38,D[illegible]30,F800,0000,039E        PSL.M[illegible]?,J/HALT.INST     ;GO TELL CONSOLE WE HIT HALT INSTR
```

```
                                          ;7598   .TOC    "       I-stream decode forks : B-FORK for VAX Instructions"
                                          ;7599
                                          ;7600   ;Control passes to this point from any 'B.FORK' state.
                                          ;7601   ;The state of the data path is :
                                          ;7602   ;         LA, LB = Register selected by bits <3:0> of IB byte 1
                                          ;7603   ;         VA     = Address of first operand
                                          ;7604   ;         D      = First operand
                                          ;7605   ;         Q      = Instruction stream data, if any
                                          ;7606   ;         PC     = Address of next specifier
                                          ;7607
                                          ;7608   200:     ;-------------------------------;
                                          ;7609   B.FORK:  Q_D,D_Q,                        ;S^# SHORT LITERAL
                                          ;7610            ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 0200, 0C00,007F,15E0,BC00,0000,0300     ;7611            C.FORK
                                          ;7612
                                          ;7613   201:     ;-------------------------------;
U 0201, 0000,003C,0180,F800,0000,0001     ;7614            J/RSVMOD                        ;RESERVED MODE
                                          ;7615
                                          ;7616   202:     ;-------------------------------;
                                          ;7617            RC[T1]_Q,                       ;QUAD/DOUBLE.  PUT FIRST WORD IN T1
                                          ;7618            ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
                                          ;7619            Q_D,D_0,                        ;MOVE OP1 TO Q, 2ND WORD OF OP2 IS 0
U 0202, 0F01,207F,15E0,BD88,0000,0300     ;7620            C.FORK
                                          ;7621
                                          ;7622   203:     ;-------------------------------;
U 0203, 0000,003C,0180,F800,0000,0001     ;7623            J/RSVMOD                        ;RESERVED MODE
                                          ;7624
                                          ;7625   204:     ;-------------------------------;
                                          ;7626            Q_D,D_LA,                       ;REGISTER. GET IT FROM LATC
                                          ;7627            ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 0204, 0800,007F,15E0,BC00,0000,0300     ;7628            C.FORK
                                          ;7629
                                          ;7630   224:     ;-------------------------------;WRITE TO REGISTER
                                          ;7631   B.WR:    R(PRN)_D,DT/INST.DEP,           ;STORE RESULT IN REGISTER
                                          ;7632            SET.CC(INST),                   ;SET CC FROM IT
                                          ;7633            CLR.IB.OPC,                     ;AND GO DO NEXT INSTRUCTION
U 0224, C001,C03C,0180,F8DC,4070,0062     ;7634            PC_PC+1,J/IRD
                                          ;7635
                                          ;7636   205:     ;-------------------------------;
U 0205, 0000,003C,0180,F800,0000,0001     ;7637            J/RSVMOD                        ;
```

```
                                        ;7638   ;B FORK SPECIFIER EVALUATION:  QUAD REGISTERS
                                        ;7639
                                        ;7640   206:    ;-------------------------------;
                                        ;7641           RC[T1]_LA,                      ;QUAD REGISTER
                                        ;7642           ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 0206, 0000,007C,15E0,BD88,0000,00CB   ;7643           Q_D                             ;GET LOW-ADDR WORD TO T1
                                        ;7644
                                        ;7645           ;-------------------------------;
                                        ;7646           D_R(PRN+1),                     ;GET SECOND PART
U 00CB, 0800,003F,0180,F860,0000,0300   ;7647           C.FORK
                                        ;7648
                                        ;7649   226:    ;-------------------------------;QUAD WRITE TO REGISTER
U 0226, 0010,0038,[illegible]00,0000,00E4   ;7650           Q_RC[T0]                        ;GET LOW ADDRESS PART
                                        ;7651
                                        ;7652           ;-------------------------------;
                                        ;7653           R(PRN)_Q,                       ;STORE LOW ADDRESS PART
U 00E4, 0001,E03C,0180,F8D8,0070,0112   ;7654           SET.CC(INST)                    ; SETTING TENTATIVE CONDITION CODE
                                        ;7655
                                        ;7656           ;-------------------------------;
                                        ;7657           R(PRN+1)_D,                     ;HIGH ADDRESS PART IS IN D
                                        ;7658           N_AMX.Z_TST,                    ;GET FINAL CC, Z IS 1 IFF BOTH ZERO
                                        ;7659           CLR.IB.OPC,                     ;FORGET THIS INSTRUCTION
U 0112, C001,003C,0180,F8E4,4030,0062   ;7660           PC_PC+1,J/IRD                   ;MOVE ON TO NEXT
                                        ;7661
                                        ;7662   207:    ;-------------------------------;
U 0207, 0000,003C,0180,F800,0000,0001   ;7663           J/RSVMOD
                                        ;7664
                                        ;7665   208:    ;-------------------------------;
                                        ;7666   B.DR:   Q&VA_LA,                        ;(R)
                                        ;7667           ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 0208, 0000,087C,15C0,BC00,0200,00D0   ;7668           DATA.TYPE?,J/B.M
                                        ;7669
                                        ;7670   209:    ;-------------------------------;
                                        ;7671           R(PRN)_LA+K[SP1.CON].RLOG,      ;UPDATE THE STACK POINTER
U 0209, 0018,0018,1580,F8D8,0000,0208   ;7672           J/B.DR                          ;THEN LOAD UN-INCREMENTED ADDR
                                        ;7673
                                        ;7674   20A:    ;-------------------------------;
                                        ;7675           R(PRN)_LA-K[SP1.CON].RLOG,      ;-(R) AUTO DECREMENT
                                        ;7676           Q&VA_ALU,                       ; USE DECREMENTED ADDR
                                        ;7677           ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 020A, 0018,0844,15C0,BCD8,0200,00D0   ;7678           DATA.TYPE?,J/B.M
                                        ;7679
                                        ;7680   20B:    ;-------------------------------;
                                        ;7681           Q_D,VA_LA,                      ;@(R)+ AUTO INCREMENT DEFERED
U 020B, 0000,007C,15E0,BC00,0200,0115   ;7682           ID_D.SYNC                       ;SEND FIRST OP OUT FOR ACCEL
                                        ;7683
                                        ;7684   ; ***********************************************
                                        ;7685   ; * Patch no. 052, PCS 020B trapped to WCS 117A *
                                        ;7686   ; ***********************************************
                                        ;7687
                                        ;7688           ;-------------------------------;
                                        ;7689           D[LONG]_CACHE,                  ;GET INDIRECT WORD
                                        ;7690           R(PRN)_LA+K[.4].RLOG,           ; WHILE UPDATING REGISTER
U 0115, 0018,0018.1180,40D8,0000,0128   ;7691           J/B.DF                          ;THEN JOIN COMMON CODE
```

```
                                               ;B FORK SPECIFIER EVALUATION: INDEX AND DISPLACEMENT MODES

                                               20C:    ;-------------------------------;
                                                       RC[T7]_LA.CTX,                  ;INDEX MODE, CONTEXT SHIFT INDEX
                                                       ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 020C, 0060,C07D,1580,BDB8,0000,047E                  CALL,J/ASPC                     ; AND GO EVALUATE BASE OPERAND ADDRESS

                                               26C:    ;-------------------------------;RETURN HERE FROM ASPC
                                                       Q&VA_D+LC,                      ;COMPUTE INDEXED ADDRESS
                                                       D_Q,                            ;RESTORE FIRST OPERAND TO D
U 026C, 0C11,0814,01C0,F800,0200,00D0                  DATA.TYPE?,J/B.M                ;GO GET THE OPERAND

                                               20D:    ;-------------------------------;
                                                       Q&VA_Q+LB.PC,                   ;D(R) DISPLACEMENT MODE.
                                                       CLR.IB.SPEC,                    ;DISCARD THE SPECIFIER
                                                       ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 020D, D005,2854,15C0,BC00,0200,00D0                  DATA.TYPE?,J/B.M                ;GO GET THE OPERAND

                                               20F:    ;-------------------------------;
                                                       Q_D,VA_Q+LB.PC,                 ;@D(R) DISPLACEMENT DEFERED
                                                       ID_D.SYNC,                      ;SEND FIRST OP OUT FOR ACCEL
U 020F, D005,2054,15E0,BC00,0200,0118                  CLR.IB.SPEC                     ;DROP THE SPECIFIER

                                               ; ***********************************************
                                               ; * Patch no. 053, PCS 020F trapped to WCS 117B *
                                               ; ***********************************************

                                                       ;-------------------------------;
U 0118, 0000,003C,0180,4000,0000,0128                  D[LONG]_CACHE                   ;GET INDIRECT, GO USE IT AS ADDR

                                                       ;-------------------------------;
                                               B.DF:   Q&VA_D,                         ;USE POINTER AS ADDRESS
                                                       D_Q,                            ;RESTORE FIRST OPERAND TO D
U 0128, 0C01,083C,01C0,F800,0200,00D0                  DATA.TYPE?,J/B.M                ;
```

```
                                          ;7726    ;HERE ARE VARIANTS OF THE B-FORK ENTRY POINTS FOR R=PC
                                          ;7727
                                          ;7728    214:     ;--------------------------------;
U 0214, 0000,003C,0180,F800,0000,0001     ;7729             J/RSVMOD                          ;PC REGISTER MODE
                                          ;7730
                                          ;7731    215:     ;--------------------------------;
U 0215, 0000,003C,0180,F800,0000,0001     ;7732             J/RSVMOD                          ;ILLEGAL REGISTER MODE. R=PC
                                          ;7733
                                          ;7734    216:     ;--------------------------------;
U 0216, 0000,003C,0180,F800,0000,0001     ;7735             J/RSVMOD                          ;PC QUAD REGISTER MODE
                                          ;7736
                                          ;7737    217:     ;--------------------------------;
U 0217, 0000,003C,0180,F800,0000,0001     ;7738             J/RSVMOD                          ;ILLEGAL QUAD REGISTER MODE, R=PC
                                          ;7739
                                          ;7740    218:     ;--------------------------------;
U 0218, 0000,003C,0180,F800,0000,0001     ;7741             J/RSVMOD                          ;(PC)
                                          ;7742
                                          ;7743    219:     ;--------------------------------;
                                          ;7744             Q_D,D_Q,                          ;(PC)+ IMMEDIATE MODE
                                          ;7745             ID_D.SYNC,                        ;SEND FIRST OP OUT FOR ACCEL
                                          ;7746             CLR.IB.SPEC,
U 0219, DC00,087C,15E0,BC00,0000,00E5     ;7747             DATA.TYPE?,J/B.I                  ; BEWARE ADDRESS SOURCES
                                          ;7748
                                          ;7749    21A:     ;--------------------------------;
U 021A, 0000,003C,0180,F800,0000,0001     ;7750             J/RSVMOD                          ;-(PC)
                                          ;7751
                                          ;7752    21B:     ;--------------------------------;
                                          ;7753             VA_Q,                             ;@(PC)+ ABSOLUTE MODE
                                          ;7754             ID_D.SYNC,                        ;SEND FIRST OP OUT FOR ACCEL
                                          ;7755             CLR.IB.SPEC,
U 021B, D001,287C,1580,BC00,0200,00D0     ;7756             DATA.TYPE?,J/B.M
                                          ;7757
                                          ;7758    21C:     ;--------------------------------;
U 021C, 0000,003C,0180,F800,0000,0001     ;7759             J/RSVMOD                          ;INDEX MODE, R=PC
                                          ;7760
                                          ;7761    21D:     ;--------------------------------;
U 021D, 0000,003C,0180,F800,0000,0001     ;7762             J/RSVMOD                          ;NESTED INDEX MODE, R=PC
                                          ;7763
                                          ;7764    21F:     ;--------------------------------;
                                          ;7765             RC[T1]_Q,                         ;QUAD IMMEDIATE
                                          ;7766             Q_IB.DATA,                        ;GET SECOND PART
                                          ;7767             CLR.IB.COND,                      ;DISCARD IT FROM IB
                                          ;7768             PC_PC+4,                          ;STEP PC OVER SECOND PART OF LITERAL
U 021F, F001,2B3C,01F0,F98E,0000,01E0     ;7769             IB.TEST?,J/B.IQ                   ;MAKE SURE IT'S ALL THERE
```

```
                                          ;7770   ;HERE OFF B-FORK WHEN INSTRUCTION DECODE ROMS INDICATE SHOULD NOT DO BFORK
                                          ;7771
                                          ;7772   287:      ;-------------------------------; SHOULD NEVER HAPPEN
U 0287, 0000,003D,0180,F800,0000,0EE0     ;7773             CALL,J/EH.USEQ                   ;GET A MACHINE CHECK
                                          ;7774
                                          ;7775   ; ***********************************************
                                          ;7776   ; * Patch no. 073, PCS 0287 trapped to WCS 1181 *
                                          ;7777   ; ***********************************************
                                          ;7778
                                          ;7779   ;HERE OFF B-FORK, WHEN INSTRUCTION BUFFER DOES NOT HAVE ENOUGH DATA
                                          ;7780
                                          ;7781   27C:      ;-------------------------------;
U 027C, 0000,003D,0180,F800,0000,0E64     ;7782             CALL,J/IB.TBM                    ;TB MISS.  REFILL IT
                                          ;7783
                                          ;7784   27D:      ;-------------------------------;
U 027D, 0000,003D,0180,F600,0000,0B80     ;7785             CALL,J/IB.ERR                    ;ANY ERROR.  FIND OUT WHAT HAPPENED
                                          ;7786
                                          ;7787   27E:      ;-------------------------------;
                                          ;7788             MCT/ALLOW.IB.READ,               ;STALL.  WAIT FOR THE DATA TO COME IN
U 027E, F000,003F,01F0,F847,0000,0200     ;7789             B.FORK                           ;
```

```
                                        ;7790   ;HERE FOR THE SECOND AND SUBSEQUENT STATES OFF B-FORK
                                        ;7791
                                        ;7792                                                   ;GET HERE BY DATA.TYPE?
                                        ;7793   =000        ;-------------------------------;WRITE DESTINATION
                                        ;7794   B.M:        ALU_D,SET.CC(INST),             ;SET CONDITION CODES ON RESULT
                                        ;7795               CACHE_D.INST.DEP,               ;STORE RESULT
                                        ;7796               CLR.IB.OPC,                     ;GO DO NEXT INSTRUCTION
U 00D0, C001,C03C,0180,3004,4070,0062   ;7797               PC_PC+1,J/IRD
                                        ;7798
                                        ;7799               ;-------------------------------;
                                        ;7800               Q_D,                            ;STORE QUAD/DOUBLE RESULT
                                        ;7801               D_RC[T0],                       ;GET LOW ADDRESS PART TO STORE FIRST
U 00D1, 0810,0038,01E0,F900,0000,0154   ;7802               J7B.WQ                          ;THEN STORE HIGH ADDRESS PART
                                        ;7803
                                        ;7804   =100        ;-------------------------------;READ OR MODIFY
                                        ;7805               RC[T7]_Q,                       ;SAVE OPERAND ADDRESS
                                        ;7806               Q_D,                            ;SAVE FIRST OPERAND IN Q
                                        ;7807               D_CACHE.INST.DEP,               ;GET NORMAL B, W, L, OR F DATA
U 00D4, 0001,E03F,01E0,59B8,0000,0300   ;7808               C.FORK                          ;GO EXECUTE
                                        ;7809
                                        ;7810               ;-------------------------------;QUAD/DOUBLE
                                        ;7811               RC[T7]_Q,                       ;OPERAND ADDRESS TO T7
                                        ;7812               Q_D,                            ;FIRST OP TO Q, MAKE ROOM FOR SECOND
                                        ;7813               D_CACHE.INST.DEP,               ;GET FIRST LONGWORD OF QUAD/DOUBLE
U 00D5, 0001,E03C,01E0,59B8,0000,012B   ;7814               J7B.MQ
                                        ;7815
                                        ;7816               ;-------------------------------;FIELD SOURCE
                                        ;7817               Q_D,D_Q,                        ;FIRST OP TO Q, ADDR OF SECOND TO D
U 00D6, 0C00,003F,01E0,F800,0000,0300   ;7818               C.FORK
                                        ;7819
                                        ;7820               ;-------------------------------;ADDRESS SOURCE
                                        ;7821               Q_D,D_Q,                        ;FIRST OP TO Q, ADDRESS TO D
U 00D7, 0C00,003F,01E0,F800,0000,0300   ;7822               C.FORK
                                        ;7823
                                        ;7824   =;END OF DATA.TYPE BRANCH
```

```
                                             ;HERE TO READ SECOND LONGWORD OF A QUAD/DOUBLE OPERAND

                                                           ;--------------------------------;
                                             B.MQ:         RC[T1]_D,                         ;STORE FIRST PART OF QUAD/DOUBLE OP
                                                           ID_D.SYNC,                        ;SEND IT TO ACCELERATOR
U 012B, 0001,007C,1580,BD8B,0000,013D                      VA_VA+4                           ;GET ADDRESS OF SECOND PART

                                                           ;--------------------------------;
                                                           D[LONG]_CACHE,                    ;GET SECOND PART OF QUAD/DOUBLE
U 013D, 0000,003F,0180,4000,0000,0300                      C.FORK                            ;GO EXECUTE WITH IT

                                             ;HERE TO STORE QUAD/DOUBLE RESULT INTO MEMORY, SETTING CONDITION CODES

                                                           ;--------------------------------;
                                             B.WQ:         CACHE_D.INST.DEP,                 ;STORE QUAD/DOUBLE RESULT
U 0154, 0001,C03C,0180,30C0,0070,0161                      ALU_D,SET.CC(INST)                ;SETUP TENTATIVE CC FROM RESULT

                                                           ;--------------------------------;
                                                           D_Q,                              ;GET HIGH-ADDRESS DATA
U 0161, 0C00,003C,0180,F803,0000,0164                      VA_VA+4                           ;AND GO WRITE IT

                                                           ;--------------------------------;
                                                           CACHE_D[LONG],                    ;STORE SECOND PART OF QUAD RESULT
                                                           ALU_D,N_AMX.Z_TST,                ; Z=1 IFF BOTH PARTS ZERO
                                                           CLR.IB.OPC,                       ;GO BACK TO IRD
U 0164, C001,003C,0180,3004,4030,0062                      PC_PC+1,J/IRD                     ;
```

```
                                         ;HERE FOR QUAD/DOUBLE I-STREAM LITERALS
                                         ;THE FIRST LONGWORD OF LITERAL IS IN T1 ALREADY, WE'VE TRIED TO READ THE
                                         ; SECOND LONGWORD, AND HERE WE TEST TO SEE IF WE GOT IT.

                                         =00       ;--------------------------------;
U 01E0, 0000,003D,0180,F800,0000,0E64    B.IQ:     CALL,J/IB.TBM                    ;ISTREAM HAD A TB MISS

                                                   ;--------------------------------;
U 01E1, 0000,003D,0180,F800,0000,0B80              CALL,J/IB.ERR                    ;I BUFFER STOPPED FOR AN ERROR

                                                   ;--------------------------------;
                                                   Q_IB.DATA,CLR.IB2-5,             ;STALL WAITING FOR THE DATA
U 01E2, F000,0B3C,01F0,F800,0000,01E0              IB.TEST?,J/B.IQ                  ;LOOP UNTIL IT ARRIVES

                                                   ;--------------------------------;
                                                   D_Q,Q_D,                         ;LEAVE IT IN D, MOVE FIRST OP TO Q
                                                   ID_D.SYNC,                       ;SEND FIRST OP OUT FOR ACCEL
                                                   CLR.IB.SPEC,                     ;GOT IT, CLEAR IT
U 01E3, DC00,087C,15E0,BC00,0000,00E5              DATA.TYPE?

                                         =101      ;--------------------------------;
U 00E5, 0000,003F,0180,F800,0000,0300    B.I:      C.FORK                           ;NORMAL SRC, GO USE IT

                                                   ;--------------------------------;
U 00E7, 0814,0038,0180,F800,0000,016A              D_PC                             ;ADDRESS SOURCE.  FIGURE OUT ADDR

                                                   ;--------------------------------;
                                                   D_D-K[SP1.CON],                  ; BY SUBTRACTING SIZE FROM CURRENT PC
U 016A, 0819,0003,1580,F800,0000,0300              C.FORK                           ;
```

```
                                             ;HERE FOR CERTAIN 3-OPERAND INTEGER AND BOOLE INSTRUCTIONS
                                             ; NAMELY ADDx3, SUBx3, BISx3, BICx3, XORx3 FOR x=B, W, L
                                             ; WITH THE SECOND SPECIFIER SHORT LITERAL, AND THE THIRD REGISTER MODE.

                                             2C0:     ;-------------------------------;
                                                      ALU_Q[INST.DEP]D,                 ;SL-R OPERATION
                                                      R(SP1)_ALU,                       ;PUT RESULT IN DEST REGISTER
                                                      SET.CC(INST),
                                                      CLR.IB0-1,                        ;DROP OPCODE & DST SPEC
U 02C0, 401D,E00C,0180,F8C5,4070,0062                 PC_PC+2,J/IRD

                                             ;HERE FOR CERTAIN 3-OPERAND INTEGER AND BOOLE INSTRUCTIONS
                                             ; NAMELY ADDx3, SUBx3, BISx3, BICx3, XORx3 FOR x=B, W, L
                                             ; WITH BOTH SECOND AND THIRD SPECIFIERS INDICATING REGISTER MODE OPERANDS.

                                             2C4:     ;-------------------------------;
                                                      ALU_LA[INST.DEP]D,                ;R-R OPERATION
                                                      R(SP1)_ALU,
                                                      SET.CC(INST),                     ;CC ARE INSTR DEPENDENT
                                                      CLR.IB0-1,
U 02C4, 401C,E00C,0180,F8C5,4070,0062                 PC_PC+2,J/IRD

                                             ;HERE FOR CERTAIN INTEGER INSTRUCTIONS WHICH DO NOT WRITE A DESTINATION,
                                             ; NAMELY BITB, BITW, BITL, CMPB, CMPW, CMPL
                                             ; WITH THE SECOND SPECIFIER REGISTER MODE.

                                             2AF:     ;-------------------------------;
                                             BIT.R:
                                             CMP.R:   ALU_LA[INST.DEP]D,                ;OPERATE REGISTER AGAINST MEM
                                                      SET.CC(INST),                     ; SET THE CONDITION CODES IN PSL
                                                      CLR.IB.OPC,                       ;DISCARD OP, DO NEXT INSTR
U 02AF, C01C,E00C,0180,F804,4070,0062                 PC_PC+1,J/IRD

                                             ;HERE FOR CERTAIN INTEGER AND BOOLE INSTRUCTIONS WHICH WRITE A DESTINATION,
                                             ; NAMELY ADDx2, SUBx2, BISx2, BICx2, XORx2
                                             ; FOR x=B, W, L, AND ADWC, SBWC
                                             ; WITH THE SECOND SPECIFIER REGISTER MODE.

                                             227:     ;-------------------------------;
                                                      ALU_LA[INST.DEP]D,                ;MEM-R OPERATION
                                                      R(PRN)_ALU,                       ;RESULT INTO DST REGISTER
                                                      SET.CC(INST),                     ;SET CONDITION CODES ACCORDINGLY
                                                      CLR.IB.OPC,                       ;GO DO NEXT INSTR
U 0227, C01C,E00C,0180,F8DC,4070,0062                 PC_PC+1,J/IRD                     ;
```

```
                                    ;7924    ;HERE FOR AOBLSS, AOBLEQ WHEN DESTINATION IS REGISTER
                                    ;7925
                                    ;7926    223:     ;-------------------------------;
                                    ;7927    AOB.R:   Q_D,                            ;SAVE LIMIT IN Q
                                    ;7928             R(PRN)_LA+K[.1].RLOG,           ;UPDATE THE DESTINATION REGISTER
                                    ;7929             D_ALU,                          ;COPY IT FOR COMPARE
U 0223, 0818,0018,05E0,F8D8,0070,018E ;7930           SET.CC(LONG)                    ;SET CONDITION CODES FROM INDEX
                                    ;7931
                                    ;7932             ;-------------------------------;
                                    ;7933             ALU_D-Q,                        ;COMPARE INDEX TO LIMIT
                                    ;7934             CLK.UBCC,                       ;SAVE RESULT FOR BRANCH
                                    ;7935             Q_IB.BDEST,                     ;GET BDEST FROM IB
                                    ;7936             PC_PC+1,                        ;STEP PC OVER IT
U 018E, 701D,0B00,01F0,F804,0010,0210 ;7937           IB.TEST?                        ;DOES IB HAVE THE DATA?
                                    ;7938
                                    ;7939    =00      ;-------------------------------;
U 0210, 0000,003D,0180,F800,0000,0E64 ;7940  AOB.R1:  CALL,J/IB.TBM                   ;IB STOPPED FOR TB MISS
                                    ;7941
                                    ;7942             ;-------------------------------;
U 0211, 0000,003D,0180,F800,0000,0B80 ;7943           CALL,J/IB.ERR
                                    ;7944
                                    ;7945             ;-------------------------------;
                                    ;7946             Q_IB.BDEST,
U 0212, 7000,0E3C,01F0,F800,0000,0210 ;7947           IB.TEST?,J/AOB.R1
                                    ;7948
                                    ;7949             ;-------------------------------;
                                    ;7950             Q_Q+PC,                         ;COMPUTE BRANCH DESTINATION
                                    ;7951             CTR.IB0-1,                      ;BUT ASSUME NO BRANCH
                                    ;7952             PC_PC+1,                        ;ADVANCE PC TO NEXT IN LINE
U 0213, 4015,3B14,01C0,F804,4000,0461 ;7953           ALU?,J/AOB.2                    ;FIND OUT WHETHER TO BRANCH
```

```
                                          ;HERE FOR BBS, BBC, BBSS, BBCS, BBSC, BBCC, BBSSI, BBCCI
                                          ; WHEN BIT IS IN A REGISTER
                                          ; D CONTAINS THE POSITION OPERAND (A LONGWORD), AND LA CONTAINS THE
                                          ; REGISTER SELECTED BY THE SECOND SPECIFIER.

                                          2A8:      ;---------------------------------;
                                          BB.R:     ALU_D.ANDNOT.K[.1F],              ;TEST POSITION FOR LESS THAN 32
                                                    CLK.UBCC,                         ;SETTING ALU Z IF SO
                                                    Q_IB.BDEST,                       ;GET BDEST TO Q
                                                    PC_PC+1,                          ;STEP PC PAST IT
U 02A8, 7019,0B24,8DF0,F804,0010,0230               IB.TEST?                          ;CHECK THAT WE GOT BDEST

                                          =00       ;---------------------------------;
U 0230, 0000,003D,0180,F800,0000,0E64     BB.R1:    CALL,J/IB.TBM

                                                    ;---------------------------------;
U 0231, 0000,003D,0180,F800,0000,0B80               CALL,J/IB.ERR

                                                    ;---------------------------------;
                                                    Q_IB.BDEST,
U 0232, 7000,0B3C,01F0,F800,0000,0230               IB.TEST?,J/BB.R1

                                                    ;---------------------------------;
                                                    CLR.IB.SPEC,                      ;DISCARD BDEST FROM IB BYTE 1
                                                    SC_D,                             ;GET BIT POSITION INTO SC
U 0233, 0001,013C,0180,F800,0082,0020               Z?                                ;CHECK THAT IT IS LESS THAN 32

                                          =0
                                          FO.ABS.20:
                                                    ;---------------------------------;ALU Z=0
U 0020, 0000,003C,0180,F800,0000,0106               J/RSVOPR                          ;POSITION .GEQ. 32, RESERVED OPERAND

                                                    ;---------------------------------;ALU Z=1 (POSITION .LSS. 32)
                                                    ALU_LA.ANDNOT.MASK,               ;TEST SELECTED BIT OF REGISTER
U 0021, 0000,0024,0180,F800,0010,0191               CLK.UBCC                          ;SET Z ACCORDINGLY

                                                    ;---------------------------------;
                                                    R(PRN)_LA[INST.DEP]MASK,          ;MODIFY THE BIT AS REQUIRED
U 0191, 0000,1B0C,0180,F8D8,0000,00E9               ALU?                              ;TEST WHETHER TO BRANCH

                                          =1001     ;---------------------------------;
U 00E9, 2015,2014,0180,F801,4200,00AB               PC&VA_Q+PC,FLUSH.IB,J/IB.FILL     ;Z=0, BBS

                                                    ;---------------------------------;
                                                    CLR.IB.OPC,                       ;Z=0, BBC
U 00EB, 0000,003C,0180,F804,4000,0062               PC_PC+1,J/IRD

                                                    ;---------------------------------;
                                                    CLR.IB.OPC,                       ;Z=1, BBS
U 00ED, 0000,003C,0180,F804,4000,0062               PC_PC+1,J/IRD

                                                    ;---------------------------------;
U 00EF, 2015,2014,0180,F801,4200,00AB               PC&VA_Q+PC,FLUSH.IB,J/IB.FILL     ;Z=1, BBC
```

```
                                        ;B-FORK EXECUTIONS

                                        ;HERE FOR TSTB, TSTW, TSTL
                                        ; THE SOURCE OPERAND IS IN D.

                                        24D:    ;-------------------------------;
                                        TST:    ALU_D,SET.CC(INST),             ;JUST SET CC FROM DATA
                                                CLR.IB.OPC,
U 024D, C001,C03C,0180,F804,4070,0062           PC_PC+1,J/IRD

                                        ;HERE FOR INCx, DECx, FOR x= B, W, L AND DESTINATION NOT REGISTER
                                        ; THE DESTINATION OPERAND IS IN D

                                        24E:    ;-------------------------------;
                                        INC:
                                        DEC:    D_D[INST.DEP]Q,                 ;OPERATE ON DATA IN D
                                                SET.CC(INST),                   ;SET CC IN PSL ACCORDINGLY
U 024E, 081D,C00C,0180,F800,0070,0341           J/STORE                         ;STORE IT ACCORDING TO VA

                                        ;HERE FOR PUSHL AND PUSHAB, PUSHAW, PUSHAL, PUSHAQ
                                        ; D CONTAINS THE SOURCE OPERAND (ADDRESS, IN THE CASE OF PUSHA)

                                        248:    ;-------------------------------;
                                        PUSHL:
                                        PUSHA:  ALU_D,SET.CC(INST),             ;SET PSL CC FROM DATA TO STORE
U 0248, 0001,C03C,0180,FA70,0070,0194           LAB_R[SP]                       ;GET SP READY TO DECREMENT

                                                ;-------------------------------;
                                                R[SP]&VA_LA-K[.4],RLOG,         ;LOAD DECREMENTED SP INTO VA
U 0194, 0018,0004,1180,FAF0,0200,0341           J/STORE

                                        ;HERE FOR JSB.  JUMP ADDRESS IS IN D

                                        28F:    ;-------------------------------;
                                        JSB:    Q_D,                            ;JUMP ADDR TO Q,
                                                D_PC,                           ;GET PC TO SAVE
U 028F, 0814,0038,01E0,FA70,0000,01A4           LAB_R[SP]                       ;GET STACK POINTER INTO LATCH

                                                ;-------------------------------;
U 01A4, 0018,0004,1180,FAF0,0200,01BC           R[SP]&VA_LA-K[.4],RLOG          ;DECREMENT SP INTO VA

                                                ;-------------------------------;
                                                CACHE_D[LONG],                  ;STORE PC ON THE STACK
U 01BC, 0000,003C,0180,3000,0000,0163           J/JMP.Q                         ;AND JUMP TO ADDR IN Q

                                        ;HERE ON JMP.  JUMP ADDRESS IS IN D

                                        28E:    ;-------------------------------;
                                        JMP:    PC&VA_D,                        ;LOAD NEW ADDRESS
                                                FLUSH.IB,
U 028E, 2001,003C,0180,F801,4200,00AB           J/IB.FILL                       ;
```

```
                                          ;8058   ;HERE FOR BLBS, BLBC
                                          ;8059   ; THE SOURCE OPERAND IS IN D
                                          ;8060
                                          ;8061   2CF:    ;---------------------------------;
                                          ;8062   BLB:    ALU_D[INST.DEP]K[.1],              ;TEST LOW BIT FOR SET OR CLR
                                          ;8063           CLK.UBCC,                          ;SET ALU Z BIT ACCORDINGLY
                                          ;8064           Q_IB.BDEST,                        ;GET BRANCH DISP
                                          ;8065           PC_PC+1,
U 02CF, 7019,080C,05F0,F804,0010,03B8     ;8066           IB.TEST?                           ;SEE IF WE GOT IT
                                          ;8067
                                          ;8068   =00     ;00-------------------------------;TB MISS
U 03B8, 0000,003D,0180,F800,0000,0E64     ;8069   BLB.1:  CALL,J/IB.TBM                      ;GO FILL IN TB, IT DOESN'T KNOW
                                          ;8070
                                          ;8071           ;01-------------------------------;IB ERROR
U 03B9, 0000,003D,0180,F800,0000,0B80     ;8072           CALL,J/IB.ERR                      ;IBUFFER STOPPED FOR AN ERROR
                                          ;8073
                                          ;8074           ;10-------------------------------;STALL
                                          ;8075           Q_IB.BDEST,                        ;WAIT FOR DATA
U 03BA, 7000,083C,01F0,F800,0000,03B8     ;8076           IB.TEST?,J/BLB.1                   ; TEST FOR ARRIVAL OF BDEST
                                          ;8077
                                          ;8078           ;---------------------------------;
                                          ;8079           D_Q+PC,                            ;COMPUTE BRANCH ADDR
                                          ;8080           CLR.IB0-1,                         ;DISCARD OPCODE IN CASE NO BRANCH
                                          ;8081           PC_PC+1,                           ;AND STEP PC TO NEXT
U 03BB, 4815,2114,0180,F804,4000,00A0     ;8082           Z?                                 ;SHOULD WE BRANCH?
                                          ;8083
                                          ;8084   =0      ;---------------------------------; Z=0
                                          ;8085           PC&VA_D,                           ;TAKE THE BRANCH
U 00A0, 2001,003C,0180,F801,4200,00AB     ;8086           FLUSH.IB,J/IB.FILL
                                          ;8087
                                          ;8088           ;--------------------------    ---; Z=1
U 00A1, F80C,003B,01F1,F857,139B,6000     ;8089           IRD                                ;DON'T BRANCH
```

```
                                        ;8090   ;HERE FOR SOBGTR, SOBGEQ WHEN DESTINATION IS NOT REGISTER
                                        ;8091   ; THE DESTINATION OPERAND IS IN D
                                        ;8092
                                        ;8093   2CE:    ;-------------------------------;
                                        ;8094   SOB:    D_D-K[.1],                      ;DECREMENT THE OPERAND
                                        ;8095           SET.CC(LONG),                   ;GET BRANCH CONDITION
                                        ;8096           Q_IB.BDEST,                     ;GET & CLR BRANCH DISPLACEMENT FROM IB
                                        ;8097           PC_PC+1,
U 02CE, 7819,0800,05F0,F804,0070,03E8   ;8098           IB.TEST?                        ;SEE IF WE GOT IT OK
                                        ;8099
                                        ;8100   =00     ;-------------------------------;
U 03E8, 0000,003D,0180,F800,0000,0E6E   ;8101   SOB.1:  CALL,J/IB.TBR
                                        ;8102
                                        ;8103           ;-------------------------------;
U 03E9, 0000,003D,0180,F800,0000,0B80   ;8104           CALL,J/IB.ERR
                                        ;8105
                                        ;8106           ;-------------------------------;
                                        ;8107           Q_IB.BDEST,                     ;WAIT FOR IBUFFER TO GET BDEST
U 03EA, 7000,0B3C,01F0,F800,0000,03E8   ;8108           IB.TEST?,J/SOB.1                ;LOOP TESTING
                                        ;8109
                                        ;8110           ;-------------------------------;
                                        ;8111           CACHE_D[LONG],                  ;STORE RESULT IN MEMORY
                                        ;8112           CLR.IB.SPEC,                    ;DISCARD BDEST FROM IB BYTE 1
                                        ;8113           Q_Q+PC,                         ;COMPUTE BRANCH ADDRESS
U 03EB, D015,3A14,01C0,3000,0000,0163   ;8114           PSL.CC?                         ;BRANCH?
                                        ;8115
                                        ;8116   =0011   ;-------------------------------;N=0, Z=0
                                        ;8117   JMP.Q:                                  ;ALSO USED BY JSB
U 0163, 2001,203C,0180,F801,4200,00AB   ;8118   SOB.2:  PC&VA_Q,FLUSH.IB,J/IB.FILL      ;RESULT GTR, SO BRANCH
                                        ;8119
                                        ;8120           ;-------------------------------;N=0, Z=1
                                        ;8121           CLR.IB.OPC,PC_PC+1,
U 0167, C000,1B3C,0180,F804,4000,012D   ;8122           IR0?,J/SOB.3                    ;BRANCH IFF SOBGEQ
                                        ;8123
                                        ;8124           ;-------------------------------;N=1, Z=0
                                        ;8125           CLR.IB.OPC,PC_PC+1,             ;RESULT LSS, DO NOT BRANCH
U 016B, C000,003C,0180,F804,4000,0062   ;8126           J/IRD
                                        ;8127
                                        ;8128   =;END PSL.CC TEST
                                        ;8129
                                        ;8130   =1101   ;-------------------------------;IR0=0
U 012D, 2001,203C,0180,F801,4200,00AB   ;8131   SOB.3:  PC&VA_Q,FLUSH.IB,J/IB.FILL      ;SOBGEQ BRANCHES ON ZERO
                                        ;8132
                                        ;8133           ;-------------------------------;IR0=1
U 012F, F80C,003B,01F1,F857,139B,6000   ;8134           IRD                             ;SOBGTR DOES NOT BRANCH ON ZERO
```

```
                                        ;8135   ;HERE FOR BISPSW AND BICPSW
                                        ;8136   ; THE SOURCE OPERAND IS IN D
                                        ;8137
                                        ;8138   28B:    ;-------------------------------;
                                        ;8139   BISPSW:
                                        ;8140   BICPSW: Q_ID[PSL],                      ;GET PSL READY FOR MODIFICATION
U 028B, 0803,403C,3DF0,2C00,0000,01D5   ;8141           D_D.OXT[WORD]                   ;DISCARD GARBAGE FROM WORD OPERAND
                                        ;8142
                                        ;8143           ;-------------------------------;
                                        ;8144           D_Q[INST.DEP]D,                 ;SETUP PSL
U 01D5, 081D,380C,0180,F800,0000,02B9   ;8145           D.B1?                           ;IS BYTE 1 CLEAR?
                                        ;8146
                                        ;8147   =01     ;0------------------------------;BYTE 1 .EQL.0
                                        ;8148           ID[PSL]_D,                      ;WRITE BACK PSL
                                        ;8149           CLR.IB.OPC,
U 02B9, C000,003C,3D80,3C04,4000,0062   ;8150           PC_PC+1,J/IRD
                                        ;8151
                                        ;8152           ;1------------------------------;BYTE 1.NEQ.0
U 02BB, 0000,003C,0180,F800,0000,0106   ;8153           J/RSVOPR                        ;TAKE RESERVED OPERAND TRAP
                                        ;8154
                                        ;8155   ;HERE FOR MNEGx, MCOMx FOR x=B, W, L
                                        ;8156   ; THE SOURCE OPERAND IS IN D
                                        ;8157
                                        ;8158   249:    ;-------------------------------;
                                        ;8159   MNEG:
                                        ;8160   MCOM:   D_Q[INST.DEP]D,                 ;OPERATE ON D
                                        ;8161           SET.CC(INST),                   ;LOAD PSL CC ON FUNCTION
U 0249, F81D,E00F,01F0,F847,0070,0300   ;8162           WRITE.DEST,J/WRD                ;GO EVALUATE DEST SPECIFIER
                                        ;8163
                                        ;8164   ;HERE FOR MOVZBW, MOVZBL, MOVZWL
                                        ;8165   ; THE SOURCE OPERAND IS IN D
                                        ;8166
                                        ;8167   24A:    ;-------------------------------;
                                        ;8168   MOVZ:   D_D.OXT[INST.DEP],              ;ZERO EXTEND THE SOURCE OPERAND TO LONG
U 024A, F803,C03F,01F0,F847,00C0,0200   ;8169           B.FORK                          ;GO WRITE THAT RESULT AS MOVE WOULD
```

```
                                          ;8170    ;HERE FOR ADAWI
                                          ;8171    ;ADD ALIGNED WORD INTERLOCKED
                                          ;8172    ; D CONTAINS SOURCE
                                          ;8173
                                          ;8174    20E:     ;-------------------------------;
                                          ;8175    ADAWI:   SC_K[ZERO],                    ;CLEAR SC FOR BRANCH ON RETURN
U 020E, 0000,003D,1980,F800,0084,647E     ;8176             CALL,J/ASPC                    ;EVALUATE DESTINATION ADDRESS
                                          ;8177
                                          ;8178    26E:     ;-------------------------------;RETURN HERE WITH MEMORY OPERAND
U 026E, 0000,003C,0180,F800,0000,00D2     ;8179             MUL?,J/ADA.1                   ;TEST ALIGNMENT OF DESTINATION
                                          ;8180
                                          ;8181    26F:     ;-------------------------------;RETURN HERE WITH REGISTER OPERAND
                                          ;8182             R(PRN)_LA+Q,                   ;PUT RESULT IN REGISTER
                                          ;8183             SET.CC(INST),
                                          ;8184             PC_PC+1,CLR.IB.OPC,            ;MOVE ON TO NEXT INSTRUCTION
U 026F, C01C,C014,0180,F8DC,4070,0062     ;8185             J/IRD
                                          ;8186
                                          ;8187    =10      ;-------------------------------;D0=0, DEST IS WORD ALIGNED
                                          ;8188    ADA.1:   D[WORD]_CACHE.LK,              ;GET DEST INTERLOCKED
U 00D2, 0000,403C,0180,7000,0000,01F3     ;8189             J/ADA.2
                                          ;8190
                                          ;8191             ;-------------------------------;D0=1, UNALIGNED
U 00D3, 0000,003C,0180,F800,0000,0106     ;8192             J/RSVOPR
                                          ;8193
                                          ;8194             ;-------------------------------;
U 01F3, 081D,C014,0180,F800,0070,01F9     ;8195    ADA.2:   D_D+Q,SET.CC(INST)             ;ADD SOURCE TO DESTINATION
                                          ;8196
                                          ;8197             ;-------------------------------;
                                          ;8198             CACHE_D[WORD].LK,              ;WRITE IT BACK
U 01F9, C000,403C,0180,3804,4000,0062     ;8199             CLR.IB.OPC,PC_PC+1,J/IRD       ;GO TO NEXT INSTRUCTION
```

```
                                        ;8200   ;FLOAT/DOUBLE EXECUTIONS WITH SINGLE OPERAND
                                        ;8201
                                        ;8202   28C:     ;--------------------------------;TSTD
                                        ;8203   TSTD:    ALU_RC[T0],SET.CC(INST),          ;SET CONDITION CODES FROM SOURCE
                                        ;8204            CHK.FLT.OPR,                      ;FAULT IF RESERVED OPERAND
U 028C, C010,C038,0180,F904,4870,0062   ;8205            CLR.IB.OPC,PC_PC+1,J/IRD          ;GO TO NEXT INSTRUCTION
                                        ;8206
                                        ;8207   28D:     ;--------------------------------;TSTF
                                        ;8208   TSTF:    ALU_D,SET.CC(INST),               ;SET CONDITION CODES FROM SOURCE
                                        ;8209            CHK.FLT.OPR,                      ;FAULT IF RESERVED OPERAND
U 028D, C001,C03C,0180,F804,4870,0062   ;8210            CLR.IB.OPC,PC_PC+1,J/IRD          ;GO TO NEXT INSTRUCTION
                                        ;8211
                                        ;8212   285:     ;--------------------------------;MOVF, MNEGF
                                        ;8213   MOVF:
                                        ;8214   MNEGF:   SC_D(EXP),                        ;GET EXPONENT FOR ZERO TEST
                                        ;8215            CHK.FLT.OPR,                      ;CATCH RESERVED OPERANDS
U 0285, 0001,183C,0180,F800,0883,01CC   ;8216            D.BYTES?,J/MOVD.2                 ;IS EXPONENT ZERO (CRUDELY)?
                                        ;8217
                                        ;8218   241:     ;---------- ---------------------;
                                        ;8219   MOVD:
                                        ;8220   MNEGD:   Q_D,                              ;SAVE LOW-ORDER FRACTION BITS
                                        ;8221            D_RC[T0],                         ;GET OUT HIGH PART WITH EXPONENT
                                        ;8222            CHK.FLT.OPR,                      ;ABORT IF RESERVED OPERAND
U 0241, 0810,0038,01E0,F900,0883,01CD   ;8223            SC_ALU(EXP),J/MOVD.3              ;CATCH EXPONENT FOR ZERO TEST
                                        ;8224
                                        ;8225   =1000                                      ;SET UP CONSTRAINT BLOCK
                                        ;8226   =1011    ;--------------------------------;SC .EQL. 0 ON 'MUL?'' TEST AT MOVD.3
                                        ;8227   MOVD.1:  Q_0,RC[T1]_K[ZERO],               ;CLEAN UP POTENTIALLY DIRTY ZERO
U 01CB, 0018,C038,19F8,F988,0070,00B9   ;8228            SET.CC(INST),J/WRQ.DST
                                        ;8229
                                        ;8230   =1100    ;--------------------------------;D<15:0> .EQL. 0
                                        ;8231   MOVD.2:  D_0,ALU_K[ZERO],                  ;RESULT IS FLOATING ZERO
                                        ;8232            SET.CC(INST),                     ;SET CONDITION CODES TO ZERO
U 01CC, FF18,C03B,19F0,F847,0070,0300   ;8233            WRITE.DEST,J/WRD
                                        ;8234
                                        ;8235   =1101    ;--------------------------------;D BYTE 1 .EQL. 0 (IF MOVF, MNEGF)
                                        ;8236   MOVD.3:  RC[T1]_Q,                         ;SAVE LOW-ORDER WHERE CFORK CAN FIND
U 01CD, 0001,2C3C,0180,F988,0000,01CB   ;8237            MUL?,J/MOVD.1                     ;IS THE EXPONENT (IN SC) ZERO?
                                        ;8238
                                        ;8239   =1110    ;--------------------------------;D BYTE 1 .NEQ. 0 IN MOVF, MNEGF
                                        ;8240            D_D[INST.DEP]K[.8000],            ;TRANSFORM SIGN IF MNEG
                                        ;8241            SET.CC(INST),                     ;SET CONDITION CODES FROM DEST
U 01CE, F819,C00F,45F0,F847,0070,0300   ;8242            WRITE.DEST,J/WRD                  ;STORE THAT
                                        ;8243
                                        ;8244   =1111    ;--------------------------------;SC .NEQ. 0 OR D BYTES 1&0 .NEQ. 0
                                        ;8245            D_D[INST.DEP]K[.8000],            ;TRANSFORM SIGN IF MNEG
                                        ;8246            SET.CC(INST),                     ;SET CONDITION CODES FROM DEST
U 01CF, F819,C00F,45F0,F847,0070,0300   ;8247            WRITE.DEST,J/WRD                  ;STORE THAT
```

```
                                          ;HERE FOR CVTBW, CVTBL, CVTWL, WITH THE SOURCE OPERAND IN D
                                          ; THEY ARE EASY, BECAUSE THEY NEED ONLY SIGN EXTEND THE SOURCE

                                          24C:    ;-------------------------------;
                                          CVTBW:
                                          CVTBL:
                                          CVTWL:  D_D.SXT[INST.DEP],              ;SIGN EXTEND THE SOURCE TO LONG
U 024C, F802,C03F,01F0,F847,0000,0200             B.FORK                          ;GO WRITE THAT AS MOVE WOULD

                                          ;HERE FOR CVTWB, CVTLB, CVTLW, WITH THE SOURCE OPERAND IN D
                                          ; THEY'RE TOUGHER, BECAUSE WE MUST CHECK FOR OVERFLOW

                                          24B:    ;-------------------------------;
                                          CVTLW:
                                          CVTLB:
                                          CVTWB:  Q_D.SXT[INST.DEP],              ;MAKE BYTE LONG
                                                  N&Z_ALU.V&C_0,                  ; SET CC ON IT (DEST RESULT)
U 024B, 0002,C03F,01C0,F800,0050,0286             SUB7SPEC,J/CVT.2                ;ADVANCE EXEC PT CTR FOR NEW CONTEXT

                                          286:    ;-------------------------------;
                                          CVT.2:  ALU_D.XOR.Q,SET.CC(INST),       ;SET V IF SOURCE DATA TYPE (IN D)
                                                                                  ; NOT EQL TO DST DATA TYPE IN Q
U 0286, F01D,C023,01F0,F847,0070,0300             WRITE.DEST,J/WRD                ; GO STORE RESULT
```

```
                                        ;8271   .TOC    "       I-stream decode forks : C-FORK for VAX Instructions"
                                        ;8272
                                        ;8273   ;Control passes here from any "C.FORK" state.
                                        ;8274   ;The state of the data path is:
                                        ;8275   ;       LA & LB = Register selected by second specifier
                                        ;8276   ;       VA      = address of second operand
                                        ;8277   ;       D       = second operand
                                        ;8278   ;       Q       = first operand
                                        ;8279   ;
                                        ;8280   ;Unlike A and B forks, specifier evaluation here is by subroutine call.
                                        ;8281   ; Instructions with two or more operands go to execution at
                                        ;8282   ; C fork.  Those which have three or more call either the SPEC or ASPC
                                        ;8283   ; subroutines to evaluate specifiers after the first two.
                                        ;8284
                                        ;8285
                                        ;8286   ;HERE FOR 2-OPERAND INTEGER INSTRUCTIONS WHICH DO NOT WRITE A DESTINATION,
                                        ;8287   ;NAMELY BITB, BITW, BITL, CMPB, CMPW, CMPL
                                        ;8288   ;WHEN THE SECOND SPECIFIER IS MEMORY OR SHORT LITERAL
                                        ;8289
                                        ;8290   349:    ;-------------------------------;
                                        ;8291   BIT:
                                        ;8292   CMP:    ALU_D[INST.DEP]Q,                       ;OPERATE ON OPERANDS
                                        ;8293           SET.CC(INST),                           ;CONDITION CODES ARE ONLY RESULT
                                        ;8294           CLR.IB.OPC,                             ;DISCARD OP, DO NEXT INSTRUCTION
U 0349, C01D,C00C,0180,F804,4070,0062   ;8295           PC_PC+1,J/IRD
                                        ;8296
                                        ;8297   ;HERE FOR 2-OPERAND INTEGER AND BOOLE INSTRUCTIONS WHICH WRITE A DESTINATION,
                                        ;8298   ; NAMELY ADDx2, SUBx2, BISx2, BICx2, XORx2, MNEGx, MOVx, MCOMx
                                        ;8299   ; FOR x = B, W, L, PLUS ADWC, SBWC
                                        ;8300   ;WHEN THE SECOND SPECIFIER IS MEMORY
                                        ;8301
                                        ;8302   34B:    ;-------------------------------;
                                        ;8303           D_D[INST.DEP]Q,                         ;DO THE OPERATION
                                        ;8304           SET.CC(INST),                           ;SET PSL<NZVC> ACCORDINGLY
U 034B, 081D,C00C,0180,F800,0070,0341   ;8305           J/STORE
                                        ;8306
                                        ;8307   ;HERE FOR 3-OPERAND INTEGER AND BOOLE INSTRUCTIONS
                                        ;8308   ; NAMELY ADDx3, SUBx3, BISx3, BICx3, XORx3 FOR x = B, W, L
                                        ;8309   ;WHEN THE SECOND SPECIFIER IS MEMORY OR SHORT LITERAL
                                        ;8310
                                        ;8311   34A:    ;-------------------------------;
                                        ;8312           D_D[INST.DEP]Q,                         ;DO THE OPERATION
                                        ;8313           SET.CC(INST),                           ;SET CONDITION CODES
U 034A, F81D,C00F,01F0,F847,0070,0300   ;8314           WRITE.DEST,J/WRD                        ;GO WRITE DESTINATION BY SPEC 3
                                        ;8315
                                        ;8316   ;HERE FOR MANY-OPERAND INSTRUCTIONS WHOSE EXECUTION STARTS ON D.FORK
                                        ;8317   ;RE-DISPATCH THERE FOR FURTHER OPERATION
                                        ;8318
                                        ;8319   3C0:    ;-------------------------------;
                                        ;8320           RC[T6]_Q,                               ;SAVE Q FOR LATER USE
                                        ;8321           ID[T9]_D,                               ;LIKEWISE D
U 03C0, 0001,203F,E580,3DB0,0000,0400   ;8322           SUB/SPEC,J/ASPC.B                       ;GO ON TO D.FORK
```

```
                                          ;HERE FOR ROTL, WITH THE COUNT IN Q AND THE SOURCE IN D
                                          34D:    ;---------------------------------;
                                          ROTL:   SC_Q,                              ;COUNT TO SC
U 034D, 0001,203C,01E0,F800,0082,01FA             Q_D                                ;REPLICATE SOURCE TO Q AND D

                                                  ;---------------------------------;
                                                  D_DAL.SC,                          ;ROTATE
U 01FA, FD00,003F,01F0,F847,0000,0200             B.FORK                             ;WRITE THE RESULT


                                          ;HERE FOR ASHL, WITH THE COUNT IN Q AND THE SOURCE IN D
                                          34C:    ;---------------------------------;
                                          ASHL:   SC_Q.SXT[BYTE],                    ;PUT COUNT IN SC
                                                  Q_0,                               ;TENTATIVE POSITIVE SIGN TO Q
                                                                                     ; IS ALSO ZEROS TO SHIFT IN LEFT
                                                  CLR.SD&SS,                         ;CLR SS FOR EASING BRANCH CONSTRAINT
U 034C, 0002,AD3C,01FF,F800,0082,0116             D31?                               ;TEST SIGN OF D

                                          =110    ;---------------------------------;D31=0
                                                  EALU_SC+K[.20],CLK.UBCC,           ;EALU N WILL SET IF SC .LSS. -32
                                                  RC[T0]_Q,                          ;REPLICATE SIGN (POS) TO T0
                                                  Q_D,D_DAL.SC,                      ;SAVE INPUT IN Q, GUESS LEFT SHIFT
U 0116, 0D01,343C,75E0,F980,0014,81E4             SC?,J/ASHL.1                       ;TEST RANGE OF SC

                                                  ;---------------------------------;D31=1
                                                  EALU_SC+K[.20],CLK.UBCC,           ;TEST FOR COUNT .LSS. -32
                                                  RC[T0]_NOT.Q,                      ;PUT NEGATIVE SIGN IN D AND T0
                                                  Q_D,D_DAL.SC,                      ;SAVE INPUT IN Q, GUESS LEFT SHIFT
U 0117, 0D01,3428,75E0,F980,0014,81E4             SC?                                ;TEST RANGE OF COUNT IN SC

                                          =100    ;---------------------------------;SC .EQL. 0
                                          ASHL.1: ALU_D,N&Z_ALU.V&C_0,               ;SET CC FROM UNMODIFIED SOURCE
U 01E4, F001,003F,01F0,F847,0050,030C             WRITE.DEST,J/WRD                   ;STORE AS RESULT

                                                  ;---------------------------------;SC .LSS. 0
                                                  D_Q,                               ;RIGHT SHIFT.  GET SRC BACK TO D
                                                  Q_RC[T0],                          ; AND GET SIGN TO Q
U 01E5, 0C10,1238,01C0,F900,0000,0096             EALU.N?,J/ASHL.4                   ;TEST FOR HUGE SHIFT AMOUNT

                                                  ;---------------------------------;0 .LSS. SC .LEQ. 31
                                                  ID[T1]_D,                          ;LEFT SHIFT.  STORE RESULT TEMPORARILY
                                                  D_RC[T0],                          ;GET SIGNS FOR OVERFLOW TEST
                                                  Q_Q.LEFT,SI/ZERO,                  ;DISCARD SIGN FROM SOURCE
U 01E6, 0810,0D38,C5A8,3D00,0000,0234             SIGNS?,J/ASHL.2                    ;BRANCH TO SET CONDITION CODE

                                                  ;---------------------------------;SC .GTR. 31
                                                  RC[T0]_0,Q_0,                      ;SOURCE MUST BE ZERO, ELSE OVERFLOW
                                                  N&Z_ALU.V&C_0,                     ; BECAUSE RESULT IS ZERO.
U 01E7, 0003,003C,01F8,F980,0050,01A9             J/ASHQ.8                           ;GO CHECK OVERFLOW
                                          =
                                          ; ***********************************************
                                          ; * Patch no. 067, PCS 01E7 trapped to WCS 118D *
                                          ; ***********************************************
```

```
                                        ;8376   ;HERE FOR LEFT SHIFT
                                        ;8377   ;RESULT IS IN ID[T1], SOURCE IS IN Q, SHIFTED LEFT ONCE,
                                        ;8378   ; 32 COPIES OF SIGN ARE IN D, AND SHIFT AMOUNT (POSITIVE) IS IN SC
                                        ;8379
                                        ;8380   =100     ;-------------------------------;D.EQL.0
                                        ;8381   ASHL.2:  ALU_0(A),LONG,N&Z_ALU.V&C_0,     ;CLEAR N, SET Z
                                        ;8382            D_DAL.SC,                        ;GATHER BITS SHIFTED INTO SIGN POSITION
U 0234, 0D03,003C,C5F0,2C00,0050,01A9   ;8383            Q_ID[T1],J/ASHQ.8                ;GET BACK RESULT FOR STORING
                                        ;8384
                                        ;8385   =110     ;-------------------------------;D.GTR.0
                                        ;8386            ALU_K[.88],LONG,N&Z_ALU.V&C_0,   ;CLEAR N & Z (CONSTANT IRRELEVANT)
                                        ;8387            D_DAL.SC,                        ;GATHER BITS SHIFTED INTO SIGN POSITION
U 0236, 0D18,0038,C5F0,2C00,0050,01A9   ;8388            Q_ID[T1],J/ASHQ.8                ;GET BACK RESULT FOR STORING
                                        ;8389
                                        ;8390            ;-------------------------------;D.LSS.0
                                        ;8391            ALU_-1,LONG,N&Z_ALU.V&C_0,       ;SET N, CLEAR Z
                                        ;8392            D_DAL.SC,                        ;GATHER BITS SHIFTED INTO SIGN POSITION
U 0237, 0D03,0028,C5F0,2C00,0050,01A9   ;8393            Q_ID[T1],J/ASHQ.8                ;GET BACK RESULT FOR STORING
                                        ;8394
                                        ;8395   =;END OF CONSTRAINT FOR BEN/SIGNS
                                        ;8396
                                        ;8397   ;HERE ON RIGHT SHIFTS
                                        ;8398   ;D CONTAINS THE SOURCE, Q IS FULL OF SIGN BITS,
                                        ;8399   ; AND SC HAS THE SHIFT COUNT, WHICH IS NEGATIVE
                                        ;8400
                                        ;8401   =011*    ;-------------------------------;EALU N=0
                                        ;8402   ASHL.4:  D_DAL.SC,                        ;SHIFT SOURCE
U 0096, 0D00,003C,0180,F800,0000,01E4   ;8403            J/ASHL.1                         ;GO SET CONDITION CODES FROM RESULT
                                        ;8404
                                        ;8405            ;-------------------------------;EALU N=1 (COUNT .LSS. -32)
                                        ;8406            D_Q,                             ;RETURN SIGN BITS AS RESULT
                                        ;8407            ALU_Q,N&Z_ALU.V&C_0,             ;SET PSL CC ACCORDINGLY
U 009E, FC01,203F,01F0,F847,0050,0300   ;8408            WRITE.DEST,J/WRD                 ;STORE
```

```
                                            ;HERE FOR ASHQ, WITH THE COUNT IN Q AND THE HIGH-ORDER LONGWORD OF SOURCE IN D
                                            ;RC[T0] AND ID[T0] ARE USED TO KEEP COPIES OF THE SIGN BITS
                                            ;RC[T1] HAS THE LOW-ORDER SOURCE, AND ID[T1] HAS THE HIGH-ORDER SOURCE

                                            343:    ;---------------------------------------;
                                            ASHQ:   SC_Q.SXT[BYTE],                         ;PUT COUNT IN SC
                                                    ID[T1]_D,                               ;SAVE HIGH-ORDER SOURCE
                                                    Q_0,                                    ;TENTATIVE POSITIVE SIGN TO Q
                                                    STATE_K[.88],                           ;SET STATE 3, NO OVERFLOW YET
                                                    CLR.SD&SS,                              ;EASE LATER EALU BRANCHES
U 0343, 0002,AD3C,C5FF,3C00,1486,6146               D31?                                    ;TEST SIGN OF D

                                            =110    ;---------------------------------------;D31=0
                                                    SC_SC+K[.20],CLK.UBCC,                  ;EALU N WILL SET IF SC .LSS. -32
                                                    RC[T0]_Q,                               ;REPLICATE SIGN (POS) TO T0
                                                    Q_D,D_0,                                ;HIGH SOURCE TO Q, SIGN TO D
U 0146, 0F01,343C,75E0,F980,0094,8264               SC?,J/ASHQ.1                            ;TEST RANGE OF SC

                                                    ;---------------------------------------;D31=1
                                                    SC_SC+K[.20],CLK.UBCC,                  ;TEST FOR COUNT .LSS. -32
                                                    Q_D,                                    ;HIGH SOURCE TO Q
                                                    D_NOT.Q,                                ;MAKE D ALL NEGATIVE SIGN BITS
                                                    RC[T0]_ALU,                             ;PUT NEGATIVE SIGN IN T0, TOO
U 0147, 0801,3428,75E0,F980,0094,8264               SC?                                     ;TEST RANGE OF COUNT IN SC

                                            =100    ;---------------------------------------;SC .EQL. 0 (NO SHIFT)
                                            ASHQ.1: D_RC[T1],                               ;LOW RESULT TO D
                                                    NEZ_ALU.V&C_0,                          ;SET Z TENTATIVELY FROM LOW PART
U 0264, 0810,0D38,0180,F908,0050,0272               SIGNS?,J/ASHQ.6                         ;SINCE THIS MAY BE A LEFT SHIFT BY A
                                                                                            ; MULTIPLE OF 32, CHECK FOR OVERFLOW

                                                    ;---------------------------------------;SC .LSS. 0 (RIGHT SHIFT)
                                            ASHQ.2: D_DAL.SC, FE_SC-K[.20],                 ;HIGH RESULT TO D (IF SC IN RANGE)
U 0265, 0D00,123C,7580,F800,0104,A0B6               EALU.N?,J/ASHQ.4                        ;TEST FOR HUGE SHIFT AMOUNT

                                                    ;---------------------------------------;0 .LSS. SC .LEQ. 31
                                                    ID[T0]_D,                               ;SAVE SIGNS AGAIN
                                                    D_RC[T1],                               ;GET LOW SOURCE
                                                    Q_0,                                    ;AND ZEROS TO SHIFT IN
U 0266, 0810,0038,C1F8,3D08,0000,026D               J/ASHQ.7                                ;GO SHIFT LEFT

                                                    ;---------------------------------------;SC .GTR. 31
                                                    ALU_D.XOR.Q,                            ;CALCULATE DIFF OF HIGH SRC FROM SIGN
                                                    CLK.UBCC.LONG,                          ;SET Z IF NO OVERFLOW
U 0267, 001D,0020,3180,F800,0094,A21E               SC_SC-K[.40]                            ;REDUCE SHIFT AMOUNT BY 32
                                                                                            ; (COMPENSATING FOR EXTRA ADD ABOVE)
                                            =;END OF SC TEST
```

```
                                        ;8456   ;HERE WITH SHIFT COUNT GREATER OR EQUAL TO 32
                                        ;8457
                                        ;8458           ;-------------------------------;
                                        ;8459           Q_D,                            ;SAVE SIGNS
                                        ;8460           D_RC[T1],                       ;GET LOW SOURCE
                                        ;8461           FE_K[.20],                      ;PREPARE TO RE-OFFSET SC
U 021E, 0810,0138,75E0,F908,0104,61C8   ;8462           Z?                              ;DID OVERFLOW OCCUR?
                                        ;8463
                                        ;8464   =0      ;-------------------------------;Z=0, OVERFLOW
U 01C8, 0003,003C,0180,F800,1408,61C9   ;8465           STATE_0(A)                      ;CLEAR STATE 3, TO INDICATE OVERFLOW
                                        ;8466
                                        ;8467           ;-------------------------------;Z=1, NO OVERFLOW
                                        ;8468           ID[T1]_D,Q_D,                   ;SAVE LOW SOURCE AS HIGH
                                        ;8469           RC[T1]_0,                       ;CLEAR LOW SOURCE
                                        ;8470           D_Q,                            ;GET SIGNS BACK INTO D
                                        ;8471           SC_SC+FE,                       ;RE-OFFSET SC BY 32
U 01C9, 0C03,143C,C5E0,3D88,0080,8264   ;8472           SC?,J/ASHQ.1                    ;GO AROUND AGAIN
                                        ;8473
                                        ;8474   ;HERE ON RIGHT SHIFTS
                                        ;8475   ; Q CONTAINS THE HIGH PART OF THE SOURCE, D HAS THE HIGH RESULT
                                        ;8476   ; SC HAS THE SHIFT COUNT, WHICH WAS ORIGINALLY NEGATIVE,
                                        ;8477   ; BUT HAS HAD 32 ADDED TO IT.  WE ARE BRANCHING ON THE SIGN OF THAT ADDITION
                                        ;8478
                                        ;8479   =011*   ;-------------------------------;EALU N=0
                                        ;8480   ASHQ.4: ID[T1]_D,                       ;SAVE HIGH RESULT
                                        ;8481           D_RC[T1], SC_FE,                ;GET LOW SOURCE, RESTORE NEG SHIFT CT
U 00B6, 0810,0038,C580,3D08,0081,0251   ;8482           J/ASHQ.5                        ;GO GET LOW RESULT
                                        ;8483
                                        ;8484           ;-------------------------------;EALU N=1 (COUNT .LSS. -32)
U 00BE, 0001,203C,0180,F988,0000,0222   ;8485           RC[T1]_Q                        ;MOVE HIGH SOURCE ONTO LOW SOURCE
                                        ;8486
                                        ;8487           ;-------------------------------;
                                        ;8488           Q_RC[T0],                       ;GET SIGNS FOR HIGH WORD
                                        ;8489           D_RC[T0],                       ; AND FOR SIGNS
                                        ;8490           SC_SC+K[.20],CLK.UBCC,          ;SEE IF COUNT IS NOW REASONABLE
U 0222, 0810,0038,75C0,F900,0094,8265   ;8491           J/ASHQ.2                        ;AND LOOP UNTIL IT IS.
```

```
                                           ;HERE ON RIGHT SHIFTS WHEN THE SHIFT COUNT HAS BEEN REDUCED TO LESS
                                           ; THAN 32.  THE HIGH RESULT HAS BEEN GENERATED AND SAVED IN ID[T1], Q HAS
                                           ; THE HIGH SOURCE, AND D HAS THE LOW SOURCE.

                                                    ;--------------------------------;
                                           ASHQ.5:  D_DAL.SC,                        ;GET LOW RESULT IN D
U 0251, 0D00,003C,C5F0,2C00,0000,0254               Q_ID[T1]                         ;AND HIGH RESULT

                                                    ;--------------------------------;
                                                    ALU_D,N&Z_ALU.V&C_0,             ;TEST LOW ORDER FOR Z
U 0254, 0001,003C,0180,F800,0050,0272               J/ASHQ.6                         ;RE-ARRANGE FOR STORAGE

                                           ;ENTER HERE WITHOUT BRANCH ON RIGHT SHIFTS.  ON SHIFTS OF 0 PLACES, COME HERE
                                           ; FROM ASHQ.1 BRANCHING ON SIGNS OF D AND Q, WHICH ARE IDENTICAL.
                                           ; ON LEFT SHIFTS BY A MULTIPLE OF 32 PLACES, ENTER FROM ASHQ.1, TESTING THE
                                           ; SIGN OF THE ORIGINAL SOURCE IN D31, AND THE SIGN OF THE SHIFTED RESULT
                                           ; IN Q31.  IF THEY ARE NOT EQUAL, WE MUST SET OVERFLOW.

                                           =010     ;--------------------------------;Q31 & D31 BOTH 0
                                           ASHQ.6:  RC[T1]_Q,                        ;SAVE HIGH PART FOR LATER STORE
                                                    N_AMX.Z_TST,                     ;SET N FROM HIGH, "AND" Z WITH LOW
U 0272, 0001,373C,0180,F988,0030,0181               STATE3?,J/ASHQ.6A                ;WAS THERE AN EARLIER OVERFLOW?

                                                    ;--------------------------------;Q31 =0, D31 =1
                                                    RC[T1]_Q,                        ;SAVE HIGH PART FOR LATER STORE
                                                    N_AMX.Z_TST,                     ;SET N FROM HIGH, "AND" Z WITH LOW
                                                    Q_D,                             ;COPY LOW ORDER LONGWORD
U 0273, 0001,203C,01E0,F988,0030,01A1               J/ASHQ.7A                        ;GO SET OVERFLOW

                                                    ;--------------------------------;Q31 =1, D31 =0
                                                    RC[T1]_Q,                        ;SAVE HIGH PART FOR LATER STORE
                                                    N_AMX.Z_TST,                     ;SET N FROM HIGH, "AND" Z WITH LOW
                                                    Q_D,                             ;COPY LOW ORDER LONGWORD
U 0276, 0001,203C,01E0,F988,0030,01A1               J/ASHQ.7A                        ;GO SET OVERFLOW

                                                    ;--------------------------------;Q31 & D31 BOTH 1
                                                    RC[T1]_Q,                        ;SAVE HIGH PART FOR LATER STORE
                                                    N_AMX.Z_TST,                     ;SET N FROM HIGH, "AND" Z WITH LOW
U 0277, 0001,373C,0180,F988,0030,0181               STATE3?,J/ASHQ.6A                ;WAS THERE AN EARLIER OVERFLOW?

                                           =0***    ;--------------------------------;STATE 3=0 (OVERFLOW EARLIER)
                                           ASHQ.6A:SET.V,                            ;NOTE OVERFLOW IN PSL
U 0181, F000,003F,01F0,F847,0020,0300               WRITE.DEST,J/WRD

                                           ; ************************************************
                                           ; * Patch no. 016, PCS 0181 trapped to WCS 1153 *
                                           ; ************************************************

                                                    ;--------------------------------;STATE 3=1 (NO OVERFLOW)
U 0189, F000,003F,01F0,F847,0000,0300               WRITE.DEST,J/WRD
```

```
                                              ;HERE FOR LEFT SHIFT
                                              ; D HAS THE LOW SOURCE, Q IS ZERO,
                                              ; AND SHIFT AMOUNT (POSITIVE) IS IN SC, WITH 32 ADDED TO IT.

                                                      ;--------------------------------;
                                              ASHQ.7: D_DAL.SC,                        ;GET LOW PART OF RESULT
                                                      Q_D,                             ;SAVE LOW SOURCE IN Q
U 026D, 0D00,003C,31E0,F800,0084,A279                 SC_SC-K[.40]                     ;MAKE SHIFT COUNT NEGATIVE

                                                      ;--------------------------------;
                                                      RC[T2]_D,                        ;SAVE LOW RESULT
                                                      N&Z_ALU.V&C_0,                   ;SET TENTATIVE Z
                                                      D_Q,                             ;LOW SOURCE TO D
U 0279, 0C01,003C,C5F0,2D90,0050,027A                 Q_ID[T1]                         ;HIGH SOURCE TO Q

                                                      ;--------------------------------;
U 027A, 0D00,003C,0180,F800,0000,0291                 D_DAL.SC                         ;GET HIGH PART OF RESULT

                                                      ;--------------------------------;
                                                      RC[T1]_D,                        ;SAVE HIGH RESULT
                                                      N_AMX.Z_TST,                     ;SET CONDITION CODES FROM THAT
                                                      D_Q,                             ;HIGH SOURCE TO D
                                                      Q_ID[T0],                        ;SIGNS TO Q
U 0291, 0C01,003C,C1F0,2D88,00B0,C298                 SC_SC+1                          ;PREPARE TO TEST BITS SHIFTED INTO 31

                                                      ;--------------------------------;
                                                      D_DAL.SC,                        ;GET BITS DISCARDED FROM HIGH WORD
                                                      Q_RC[T2],                        ;GET BACK LOW RESULT
U 0298, 0D10,1738,01C0,F910,0000,01A1                 STATE3?                          ;DID WE SEE OVERFLOW IN LONG SHIFT?

                                              =0***   ;--------------------------------;STATE3=0, LONG SHIFT SAW OVERFLOW
                                              ASHQ.7A:D_Q,                             ;READY LOW RESULT
                                                      SET.V,                           ;NOTE OVERFLOW
U 01A1, FC00,003F,01F0,F847,0020,0300                 WRITE.DEST.J/WRD                 ;STORE IT BY FINAL SPECIFIER

                                              ; ***********************************************
                                              ; * Patch no. 017, PCS 01A1 trapped to WCS 1154 *
                                              ; ***********************************************

                                                      ;--------------------------------;STATE3=1, NO OVERFLOW DETECTED
                                              ASHQ.8: ALU_D.XOR.RC[T0],                ;WERE BITS DISCARDED ALL SIGNS?
                                                      SET.CC(INST),                    ;SET V IF NOT
                                                      D_Q,                             ;READY LOW RESULT FOR STORING
U 01A9, 0C11,C020,0180,F900,0070,037E                 J/SPEC                           ;GO STORE ACCORDING TO SPECIFIER
```

```
                                    ;8586   ;HERE FOR AOBLSS, AOBLEQ
                                    ;8587   ; WITH THE LIMIT IN Q, AND THE INDEX IN D
                                    ;8588
                                    ;8589   346:     ;--------------------------------;
                                    ;8590   AOB:     D_D+K[.1],                        ;INCREMENT THE INDEX
U 0346, 0819,C014,0580,F800,0070,029B ;8591          SET.CC(INST)                      ;SET CC FROM IT
                                    ;8592
                                    ;8593            ;--------------------------------;
                                    ;8594            ALU_D-Q,                          ;COMPARE INDEX TO LIMIT
                                    ;8595            CLK.UBCC,                         ;PREPARE TO BRANCH ON RESULT
                                    ;8596            Q_IB.BDEST,                       ;GET BDEST
                                    ;8597            PC_PC+1,
U 029B, 701D,0800,01F0,F804,0010,0458 ;8598          IB.TEST?
                                    ;8599
                                    ;8600   =00      ;--------------------------------;
U 0458, 0000,003D,0180,F800,0000,0E6E ;8601 AOB.1:   CALL,J/IB.TBR
                                    ;8602
                                    ;8603            ;--------------------------------;
U 0459, 0000,003D,0180,F800,0000,0B80 ;8604          CALL,J/IB.ERR
                                    ;8605
                                    ;8606            ;--------------------------------;
                                    ;8607            Q_IB.BDEST,
U 045A, 7000,0B3C,01F0,F800,0000,0458 ;8608          IB.TEST?,J/AOB.1
                                    ;8609
                                    ;8610            ;--------------------------------;
                                    ;8611            CACHE_D[LONG],                    ;STORE INDEX, UPDATED
                                    ;8612            Q_Q+PC,                           ;COMPUTE BRANCH DESTINATION
                                    ;8613            CLR.IB0-1,                        ;DISCARD THIS OPCODE & BDEST
                                    ;8614            PC_PC+1,                          ;AND POINT PC TO NEXT
U 045B, 4015,3B14,01C0,3004,4000,0461 ;8615          ALU?                              ;SHOULD WE BRANCH?
                                    ;8616
                                    ;8617   =;END OF IB.TEST
                                    ;8618
                                    ;8619   =0001    ;--------------------------------;N=0, Z=0, IR0=0
U 0461, F80C,003B,01F1,F857,139B,6000 ;8620 AOB.2:   IRD                               ;DO NOT BRANCH
                                    ;8621
                                    ;8622            ;--------------------------------;N=0, Z=0, IR0=1
U 0463, F80C,003B,01F1,F857,139B,6000 ;8623          IRD                               ;DO NOT BRANCH
                                    ;8624
                                    ;8625            ;--------------------------------;N=0, Z=1, IR0=0 (AOBLSS)
U 0465, F80C,003B,01F1,F857,139B,6000 ;8626          IRD                               ;DO NOT BRANCH
                                    ;8627
                                    ;8628            ;--------------------------------;N=0, Z=1, IR0=1 (AOBLEQ)
                                    ;8629            PC&VA_Q,FLUSH.IB,                 ;BRANCH
U 0467, 2001,203C,0180,F801,4200,00AB ;8630          J/IB.FILL
                                    ;8631
                                    ;8632            ;--------------------------------;N=1
                                    ;8633            PC&VA_Q,FLUSH.IB,                 ;BRANCH
U 0469, 2001,203C,0180,F801,4200,00AB ;8634          J/IB.FILL
                                    ;8635
                                    ;8636            ;--------------------------------;N=1
                                    ;8637            PC&VA_Q,FLUSH.IB,                 ;BRANCH
U 046B, 2001,203C,0180,F801,4200,00AB ;8638          J/IB.FILL
                                    ;8639
                                    ;8640   =;END OF ALU.CC TEST
```

```
                                            ;HERE FOR ACBB, ACBW, ACBL
                                            ; FIRST OPERAND (LIMIT) IS IN Q, SECOND OPERAND (ADDEND) IS IN D

                                            3C4:    ;-------------------------------;CALL SITE FOR INDEX SPECIFIER EVAL
                                            ACB:    RC[T0]_Q.SXT[INST.DEP],         ;SAVE LIMIT, SIGN EXTENDED
U 03C4, 0002,E03D,0180,F980,0000,037E               CALL,J/SPEC                     ;GO EVALUATE INDEX SPECIFIER

                                            3D4:    ;-------------------------------;RETURN HERE WITH MEMORY OPERAND
                                                    RC[T1]_Q.SXT[INST.DEP],         ;SAVE ADDEND, SIGN EXTENDED
                                                    Q_IB.BDEST,                     ;GET BDEST FROM IB
                                                    PC_PC+2,                        ;STEP PC PAST WORD DISPLACEMENT
U 03D4, 7002,EB3D,01F0,F98D,0000,0478               IB.TEST?,CALL,J/ACB.8           ;GO WAIT FOR IT IF NECESSARY

                                            3D5:    ;-------------------------------;INDEX IS IN MEMORY
                                                    D_D+LC,                         ;INDEX + ADDEND
                                                    CLR.IB.SPEC,                    ;DISCARD 2ND BYTE OF BDEST
                                                    SET.CC(INST),                   ;SET PSL CC FROM SUM
U 03D5, D811,C014,0180,F800,0070,02B0               J/ACB.1                         ;GO STORE BACK IN MEMORY

                                            3D6:    ;-------------------------------;RETURN HERE WITH REGISTER OPERAND
                                                    RC[T1]_Q.SXT[INST.DEP],         ;SAVE ADDEND, SIGN EXTENDED
                                                    Q_IB.BDEST,                     ;GET BDEST FROM IB
                                                    PC_PC+2,                        ;STEP PC PAST WORD DISPLACEMENT
U 03D6, 7002,EB3D,01F0,F98D,0000,0478               IB.TEST?,CALL,J/ACB.8           ;GO WAIT FOR IT IF NECESSARY

                                            3D7:    ;-------------------------------;INDEX IS IN A REGISTER
                                                    D_D+LC,                         ;INDEX + ADDEND
                                                    CLR.IB.SPEC,                    ;DISCARD 2ND BYTE OF BDEST
                                                    R(PRN)_ALU,                     ;STORE AS FINAL RESULT
U 03D7, D811,C014,0180,F8D8,0070,02A9               SET.CC(INST)

                                            =;END OF SUBROUTINE CONSTRAINT STARTED AT ACB

                                                    ;-------------------------------;
                                                    LC_RC[T0],                      ;RECOVER LIMIT
U 02A9, 0000,1B3C,0180,F900,0000,00C7               ALU.N?,J/ACB.2                  ;TEST SIGN OF ADDEND

                                                    ;-------------------------------;
                                            ACB.1:  CACHE_D[INST.DEP],              ;STORE UPDATED INDEX
                                                    LC_RC[T0],                      ;GET LIMIT BACK
U 02B0, 0000,DB3C,0180,3100,0000,00C7               ALU.N?                          ;TEST SIGN OF ADDEND

                                            =0111   ;-------------------------------;ADDEND .GEQ. 0
                                            ACB.2:  ALU_D-LC-1,                     ;COMPARE INDEX TO LIMIT
                                                    DT/INST.DEP,CLK.UBCC,           ;RESULT OF COMPARE TO ALU N
U 00C7, 0011,C008,0180,F800,0010,02D0               J/ACB.3                         ;GO TEST IT

                                            ; ***********************************************
                                            ; * Patch no. 060, PCS 00C7 trapped to WCS 1186 *
                                            ; ***********************************************

                                                    ;-------------------------------;ADDEND .LSS. 0
                                                    ALU_D-LC,                       ;COMPARE INDEX TO LIMIT
U 00CF, 0011,C000,0180,F800,0010,02C6               DT/INST.DEP,CLK.UBCC            ;
```

```
                                        ;8695   ;HERE FOR ACB, AFTER COMPARING THE UPDATED INDEX TO THE LIMIT
                                        ;8696
                                        ;8697           ;-------------------------------;ADDEND IS NEGATIVE
                                        ;8698           Q_Q+PC,                         ;CALCULATE BRANCH ADDRESS
                                        ;8699           CLR.IB.OPC,                     ;DISCARD OPCODE
                                        ;8700           PC_PC+1,                        ;AND MOVE PC TO NEXT
U 02C6, C015,3B14,01C0,F804,4000,0367   ;8701           ALU.N?                          ;TEST COMPARISON OF INDEX WITH LIMIT
                                        ;8702
                                        ;8703   =0111   ;-------------------------------;ALU.N=0 (INDEX .GEQ. LIMIT)
U 0367, 2001,203C,0180,F801,4200,00AB   ;8704           PC&VA_Q,FLUSH.IB,J/IB.FILL      ;BRANCH
                                        ;8705
                                        ;8706           ;-------------------------------;ALU N=1 (INDEX .LSS. LIMIT)
U 036F, F80C,003B,01F1,F857,139B,6000   ;8707           IRD                             ;NO BRANCH
                                        ;8708
                                        ;8709
                                        ;8710           ;-------------------------------;ADDEND IS POSITIVE
                                        ;8711   ACB.3:  Q_Q+PC,                         ;CALCULATE BRANCH ADDRESS
                                        ;8712           CLR.IB.OPC,                     ;DISCARD OPCODE
                                        ;8713           PC_PC+1,
U 02D0, C015,3B14,01C0,F804,4000,0497   ;8714           ALU.N?                          ;TEST COMPARISON OF INDEX WITH LIMIT
                                        ;8715
                                        ;8716   =0111   ;-------------------------------;ALU.N=0 (INDEX .GTR. LIMIT)
U 0497, F80C,003B,01F1,F857,139B,6000   ;8717           IRD                             ;NO BRANCH
                                        ;8718
                                        ;8719           ;-------------------------------;ALU N=1 (INDEX .LEQ. LIMIT)
U 049F, 2001,203C,0180,F801,4200,00AB   ;8720           PC&VA_Q,FLUSH.IB,J/IB.FILL      ;BRANCH
                                        ;8721
                                        ;8722
                                        ;8723   ;HERE IS SUBROUTINE USED BY ACB TO GET BRANCH DISPLACEMENT
                                        ;8724
                                        ;8725   =00     ;-------------------------------;
U 0478, 0000,003D,0180,F800,0000,0E6E   ;8726   ACB.8:  CALL,J/IB.TBR
                                        ;8727
                                        ;8728           ;-------------------------------;
U 0479, 0000,003D,0180,F800,0000,0B80   ;8729           CALL,J/IB.ERR
                                        ;8730
                                        ;8731           ;-------------------------------;
                                        ;8732           Q_IB.BDEST,                     ;TRY AGAIN TO GET BDEST
U 047A, 7000,0B3C,01F0,F800,0000,0478   ;8733           IB.TEST?,J/ACB.8                ;LOOP WAITING
                                        ;8734
                                        ;8735           ;-------------------------------;
                                        ;8736           ALU_RC[T1],CLK.UBCC,            ;GET ADDEND TO LC, TEST ITS SIGN
                                        ;8737           CLR.IB.SPEC,                    ;DISCARD 1ST BYTE OF BDEST
U 047B, D010,003A,0180,F908,0010,0001   ;8738           RETURN1
```

```
                                       ;HERE FOR BBS, BBC, BBSS, BBCS, BBSC, BBCC, BBSSI, BBCCI
                                       ; WHEN BIT TESTED IS NOT IN A REGISTER
                                       ; D CONTAINS THE BASE ADDRESS, Q HAS THE BIT POSITION

                                       345:    ;-------------------------------;
U 0345, 0001,003C,0180,F980,0000,02EE  BB:     RC[T0]_D                        ;SAVE BASE OF BIT TABLE

                                               ;-------------------------------;
                                               SC_Q,                           ;LOW BITS OF POSITION TO SC
                                               D_Q.RIGHT2,                     ;DISCARD BITS OF POSITION
                                               Q_IB.BDEST,
                                               PC_PC+1,
U 02EE, 7881,2B3C,01F0,F804,0082,04A0          IB.TEST?

                                       ; **********************************************
                                       ; * Patch no. 062, PCS 02EE trapped to WCS 1188 *
                                       ; **********************************************

                                       =00     ;-------------------------------;
U 04A0, 0000,003D,0180,F800,0000,0E64  BB.1:   CALL,J/IB.TBM

                                               ;-------------------------------;
U 04A1, 0000,003D,0180,F800,0000,0B80          CALL,J/IB.ERR

                                               ;-------------------------------;
                                               Q_IB.BDEST,
U 04A2, 7000,0B3C,01F0,F800,0000,04A0          IB.TEST?,J/BB.1

                                               ;-------------------------------;
                                               CLR.IB.SPEC,                    ;DISCARD BDEST FROM IB BYTE1
                                               D_D.RIGHT,                      ;POSITION NOW RIGHT 3 PLACES
                                               A[U_Q.OXT[BYTE], CLK.UBCC,      ;TEST BDEST FOR 0 (HACK CASE)
                                               LC_RC[T0],                      ;READY BASE ADDRESS
U 04A3, D603,A03C,7180,F900,0094,42F1          SC_SC.ANDNOT.K[.FFF8]           ;GET ONLY BIT POSITION IN BYTE

                                       =;END OF IB.TEST

                                       FO.PA.62:
                                               ;-------------------------------;
                                               VA_D+LC,                        ;COMPUTE BYTE ADDRESS OF BIT TO TEST
U 02F1, 0011,0914,0180,F800,0200,04A8          IR2-1?                          ;WHAT KIND OF REFERENCE?

                                       =*00    ;-------------------------------;NO MODIFICATION
U 04A8, 0000,813C,0180,4000,0000,01D0          D[BYTE]_CACHE, Z?, J/BB.2

                                               ;-------------------------------;SET
U 04A9, 0000,813C,0180,5000,0000,01D0          D[BYTE]_CACHE.WCHK, Z?, J/BB.2

                                               ;-------------------------------;CLEAR
U 04AA, 0000,813C,0180,5000,0000,01D0          D[BYTE]_CACHE.WCHK, Z?, J/BB.2

                                               ;-------------------------------;INTERLOCK
U 04AB, 0000,813C,0180,7000,0000,01D0          D[BYTE]_CACHE.LK, Z?, J/BB.2    ;
```

```
                                          ;8792   ;HERE FOR BB GROUP, WITH BYTE TO BE TESTED IN D,
                                          ;8793   ; AND BIT NUMBER OF BIT TO TEST IN SC
                                          ;8794
                                          ;8795   =0        ;-------------------------------;BDEST .NE. 0
                                          ;8796   BB.2:     ALU_D.ANDNOT.MASK,              ;TEST SELECTED BIT
                                          ;8797             CLK.UBCC,
U 01D0, 0001,0924,0180,F800,0010,04C8     ;8798             IR2-1?, J/BB.3
                                          ;8799
                                          ;8800             ;-------------------------------;BDEST = 0
                                          ;8801             D_D[INST.DEP]MASK,              ;DON'T TEST - JUST ALTER
                                          ;8802             C[R.IB.OPC, PC_PC+1,            ;GET RID OF OPCODE SPECIFIER
U 01D1, C801,090C,0180,F804,4000,04B8     ;8803             IR2-1?                          ;CHECK REFERENCE TYPE
                                          ;8804
                                          ;8805   =*00      ;-------------------------------;NO MODIFICATION
U 04B8, F80C,003B,01F1,F857,139B,6000     ;8806             IRD                             ;RATHER POINTLESS INSTRUCTION, WHAT?
                                          ;8807
                                          ;8808             ;-------------------------------;SET
U 04B9, 0000,803C,0180,3000,0000,0062     ;8809             CACHE_D[BYTE], J/IRD            ;WRITE BYTE AND FALL THROUGH
                                          ;8810
                                          ;8811             ;-------------------------------;CLEAR
U 04BA, 0000,803C,0180,3000,0000,0062     ;8812             CACHE_D[BYTE], J/IRD            ;WRITE BYTE AND FALL THRU
                                          ;8813
                                          ;8814             ;-------------------------------;INTERLOCK
U 04BB, 0000,803C,0180,3800,0000,0062     ;8815             CACHE_D[BYTE].LK, J/IRD         ;WRITE BYTE, RELEASE INTLK & FALL THRU
                                          ;8816
                                          ;8817
                                          ;8818   =*00      ;-------------------------------;NO MODIFICATION
                                          ;8819   BB.3:     Q_Q+PC,                         ;COMPUTE BRANCH ADDR
                                          ;8820             C[R.IB.OPC,                     ;DISCARD OPCODE SO NEXT IS READY
                                          ;8821             PC_PC+1,                        ;POINT PC PAST IT
U 04C8, C015,3B14,01C0,F804,4000,02E9     ;8822             ALU?,J/BB.5                     ;DECIDE WHETHER TO BRANCH
                                          ;8823
                                          ;8824             ;-------------------------------;SET
                                          ;8825             D_D[INST.DEP]MASK,              ;DO IT TO IT
U 04C9, 0801,000C,0180,F800,0000,0322     ;8826             J/BB.4
                                          ;8827
                                          ;8828             ;-------------------------------;CLEAR
                                          ;8829             D_D[INST.DEP]MASK,              ;DO IT TO IT
U 04CA, 0801,000C,0180,F800,0000,0322     ;8830             J/BB.4
                                          ;8831
                                          ;8832             ;-------------------------------;INTERLOCK
U 04CB, 0801,000C,0180,F800,0000,02FD     ;8833             D_D[INST.DEP]MASK
                                          ;8834
                                          ;8835             ;-------------------------------;
                                          ;8836             CACHE_D[BYTE].LK,               ;WRITE BACK, CLEARING THE LOCK
                                          ;8837             Q_Q+PC,
                                          ;8838             C[R.IB.OPC,
                                          ;8839             PC_PC+1,
U 02FD, C015,BB14,01C0,3804,4000,02E9     ;8840             ALU?,J/BB.5
                                          ;8841
                                          ;8842             ;-------------------------------;
                                          ;8843   BB.4:     CACHE_D[BYTE],                  ;WRITE BACK
                                          ;8844             Q_Q+PC,
                                          ;8845             C[R.IB.OPC,
                                          ;8846             PC_PC+1,
U 0322, C015,BB14,01C0,3004,4000,02E9     ;8847            ALU?
                                          ;8848
                                          ;8849    =1001   ;------------------------------;Z=0, BBS
U 02E9, 2001,203C,0180,F801,4200,00AB     ;8850    BB.5:   PC&VA_Q,FLUSH.IB,J/IB.FILL     ;BRANCH
                                          ;8851
                                          ;8852            ;------------------------------;Z=0, BBC
U 02EB, F80C,003B,01F1,F857,139B,6000     ;8853            IRD
                                          ;8854
                                          ;8855            ;------------------------------;Z=1, BBS
U 02ED, F80C,003B,01F1,F857,139B,6000     ;8856            IRD
                                          ;8857
                                          ;8858            ;------------------------------;Z=1, BBC
U 02EF, 2001,203C,0180,F801,4200,00AB     ;8859            PC&VA_Q,FLUSH.IB,J/IB.FILL
```

```
                                         ;8860  ;HERE FOR CASEB AND CASEW
                                         ;8861  ; THE FIRST OPERAND (SELECTOR) IS IN Q, THE SECOND (BASE) IS IN D
                                         ;8862
                                         ;8863  383:    ;-------------------------------;CALL SITE FOR LIMIT SPECIFIER EVAL
                                         ;8864  CASEB:
                                         ;8865  CASEW:  D_Q-D,                          ;COMPUTE TMP AS SELECTOR-BASE
U 0383, 081D,2001,0180,F800,0000,037E    ;8866          CALL,J/SPEC                     ;GO EVALUATE LIMIT SPECIFIER
                                         ;8867
                                         ;8868  393:    ;-------------------------------;RETURN HERE FROM SPECIFIER EVALUATION
                                         ;8869          ALU_D-Q-1,                      ;COMPARE TMP TO LIMIT
U 0393, 001D,C008,0180,F800,0070,0330    ;8870          SET.CC(INST)                    ;SET PSL CC ACCORDINGLY
                                         ;8871
                                         ;8872          ;-------------------------------;
                                         ;8873          Q_Q.OXT[INST.DEP].LEFT,         ;MULTIPLY TMP BY 2 FOR ADDRESSING BDEST
U 0330, 0023,FA3C,01C0,F800,0000,0028    ;8874          PSL.CC?                         ;DECIDE WHETHER TO BRANCH
                                         ;8875
                                         ;8876  =10*0
                                         ;8877  FO.ABS.28:
                                         ;8878          ;-------------------------------;Z=0, C=0 (TMP GTRU LIMIT)
                                         ;8879          Q_D.OXT[INST.DEP]+K[.1].LEFT,   ;(LIMIT+1)*2 GIVES LENGTH OF BDEST LIST
U 0028, 003B,C014,05C0,F800,0000,00CE    ;8880          J/BR                            ;BRANCH PAST THE LIST
                                         ;8881
                                         ;8882          ;-------------------------------;Z=0, C=1 (TMP LSSU LIMIT)
U 0029, 0015,2014,0180,F800,0200,013C    ;8883          VA_Q+PC,J/CASE.1                ;GET ADDRESS OF SELECTED BDEST
                                         ;8884
                                         ;8885  FO.ABS.2C:
                                         ;8886          ;-------------------------------;Z=1, C=0 (TMP EQL LIMIT)
U 002C, 0015,2014,0180,F800,0200,013C    ;8887          VA_Q+PC,J/CASE.1
                                         ;8888
                                         ;8889  =;END OF PSL.CC TEST
```

```
                                        ;8890   ;HERE FOR CASEL
                                        ;8891   ; THE FIRST OPERAND (SELECTOR) IS IN Q, THE SECOND (BASE) IS IN D
                                        ;8892
                                        ;8893   30E:     ;--------------------------------;CALL SITE FOR LIMIT SPEC EVALUATION
                                        ;8894   CASEL:   D_Q-D,                           ;COMPUTE TMP AS SELECTOR-BASE
U 030E, 081D,2001,0180,F800,0000,037E   ;8895            CALL,J/SPEC                      ;GO EVALUATE LIMIT SPECIFIER
                                        ;8896
                                        ;8897   31E:     ;--------------------------------;RETURN HERE FROM SPECIFIER EVALUATION
                                        ;8898            ALU_D-Q-1,                       ;COMPARE TMP TO LIMIT
                                        ;8899            SET.CC(INST),                    ;SET PSL CC ACCORDINGLY
U 031E, 001D,C008,01A8,F800,0070,036B   ;8900            Q_Q.LEFT                         ;SHIFT TMP FOR ADDRESSING BDEST LIST
                                        ;8901
                                        ;8902            ;--------------------------------;
                                        ;8903            VA_Q+PC,                         ;ADDRESS SELECTED BDEST
U 036B, 0015,3A14,0180,F800,0200,0138   ;8904            PSL.CC?                          ;DECIDE HOW TO BRANCH
                                        ;8905
                                        ;8906   =10*0    ;--------------------------------;Z=0, C=0 (TMP GTRU LIMIT)
                                        ;8907            Q_D+K[.1].LEFT,                  ;(LIMIT+1)*2 GIVES LENGTH OF BDEST LIST
U 0138, 0039,0014,05C0,F800,0000,00CE   ;8908            J/BR                             ;BRANCH PAST THE LIST
                                        ;8909
                                        ;8910            ;--------------------------------;Z=0, C=1 (TMP LSSU LIMIT)
                                        ;8911            D[WORD]_CACHE,                   ;GET SELECTED BDEST
U 0139, 0000,403C,0180,4000,0000,036D   ;8912            J/CASE.2
                                        ;8913
                                        ;8914            ;--------------------------------;Z=1, C=0 (TMP EQL LIMIT)
U 013C, 0000,403C,0180,4000,0000,036D   ;8915   CASE.1:  D[WORD]_CACHE                    ;GET SELECTED BDEST
                                        ;8916
                                        ;8917   =;END OF PSL.CC TEST
                                        ;8918
                                        ;8919            ;--------------------------------;
                                        ;8920   CASE.2:  PC&VA_D.SXT[WORD]+PC,            ;BRANCH TO SELECTED ROUTINE
U 036D, 2016,4014,0180,F801,4200,00AB   ;8921            FLUSH.IB,J/IB.FILL
```

```
                                            ;8922   .TOC    "       I-stream decode forks : C-FORK Specifier Evaluation Subroutine"
                                            ;8923
                                            ;8924   ;Control passes to this point by "CALL,J/SPEC" or from any 'WRITE.DEST" state.
                                            ;8925   ;LA, LB = Register selected by bits <3:0> of IB byte 1
                                            ;8926   ;D = Result to be stored, if WRITE.DEST; otherwise returned in Q
                                            ;8927   ;Q = Instruction stream data, if any
                                            ;8928   ;If the specifier evaluated is of 'READ" or 'MODIFY" type, control returns
                                            ;8929   ; at the call site ored with 10, for literal and memory operands, or the
                                            ;8930   ; call site ored with 12 for register operands.  If 'WRITE" type, control
                                            ;8931   ; passes to IRD to perform the next instruction.
                                            ;8932
                                            ;8933   300:    ;-------------------------------;
                                            ;8934   WRD:
                                            ;8935   C.FORK: D_Q,Q_D,                        ;SHORT LITERAL
U 0300, 0C00,003E,01E0,F800,0000,0010       ;8936           RETURN10                        ;RETURN LITERAL IN D
                                            ;8937
                                            ;8938   301:    ;-------------------------------;
U 0301, 0000,003C,0180,F800,0000,0001       ;8939           J/RSVMOD                        ;RESERVED MODE
                                            ;8940
                                            ;8941   302:    ;-------------------------------;
                                            ;8942           RC[T2]_Q,                       ;QUAD/DOUBLE SHORT LITERAL
                                            ;8943           Q_D,D_0,                        ;RETURN OLD D IN Q, REST IS ZERO
U 0302, 0F01,203E,01E0,F990,0000,0010       ;8944           RETURN10
                                            ;8945
                                            ;8946   303:    ;-------------------------------;
U 0303, 0000,003C,0180,F800,0000,0001       ;8947           J/RSVMOD                        ;RESERVED MODE
                                            ;8948
                                            ;8949   304:    ;-------------------------------;
                                            ;8950           Q_D,D_LA,                       ;REGISTER, TO READ
U 0304, 0800,003E,01E0,F800,0000,0012       ;8951           RETURN12
                                            ;8952
                                            ;8953   324:    ;-------------------------------;
                                            ;8954   WRD.R:  R(PRN)_D,DT/INST.DEP,           ;REGISTER, TO WRITE
                                            ;8955           CLR.IB.OPC,                     ;GO TO NEXT INSTR
U 0324, C001,C03C,0180,F8DC,4000,0062       ;8956           PC_PC+1,J/IRD
                                            ;8957
                                            ;8958   305:    ;-------------------------------;
U 0305, 0000,003C,0180,F800,0000,0001       ;8959           J/RSVMOD
```

```
                                          ;GENERAL SPECIFIER (C-FORK) EVALUATION, CONTINUED

                                          306:    ;-------------------------------;
                                                  RC[T2]_LA,                      ;QUAD REGISTER, TO READ
U 0306, 0000,003C,01E0,F990,0000,036E             Q_D

                                          F0.ABS.36E:
                                                  ;-------------------------------;
                                                  D_R(PRN+1),                     ;HIGH ADDRESS PART TO D
U 036E, 0800,003E,0180,F860,0000,0012             RETURN12

                                          326:    ;-------------------------------;
U 0326, 0001,003C,0180,F8D8,0000,0375             R(PRN)_D                        ;QUAD REGISTER. STUFF LOW ADDRESS PART

                                                  ;-------------------------------;
U 0375, 0810,0038,0180,F908,0000,0392             D_RC[T1]                        ;GET HIGH ADDRESS PART

                                                  ;-------------------------------;
                                                  R(PRN+1)_D,                     ;NOW STORE HIGH ADDRESS PART
                                                  CLR.IB.OPC,
U 0392, C001,003C,0180,F8E4,4000,0062             PC_PC+1,J/IRD

                                          307:    ;-------------------------------;
U 0307, 0000,003C,0180,F800,0000,0001             J/RSVMOD

                                          308:    ;-------------------------------;
                                          C.DR:   VA_LA,                          ;(R)
                                                  Q_D,
U 0308, 0000,083C,01E0,F800,0200,02D2             DATA.TYPE?,J/C.M

                                          309:    ;-------------------------------;
                                                  R(PRN)_LA+K[SP1.CON].RLOG,      ;(R)+ UPDATE THE STACK POINTER
U 0309, 0018,0018,1580,F8D8,0000,0308             J/C.DR                          ;THEN LOAD UN-INCREMENTED ADDR

                                          30A:    ;-------------------------------;
                                                  R(PRN)_LA-K[SP1.CON].RLOG,      ;-(R) AUTO DECREMENT
                                                  VA_ALU,                         ; USE DECREMENTED ADDR
                                                  Q_D,
U 030A, 0018,0804,15E0,F8D8,0200,02D2             DATA.TYPE?,J/C.M

                                          30B:    ;-------------------------------;
                                                  VA_LA,                          ;@(R)+ AUTO INCREMENT DEFERED
U 030B, 0000,003C,01E0,F800,0200,0394             Q_D                             ;SAVE DATA WHILE GETTING INDIRECT

                                          ; ************************************************
                                          ; * Patch no. 054, PCS 030B trapped to WCS 117C *
                                          ; ************************************************

                                          F[illegible]BS.394:
                                                  ;-------------------------------;
                                                  D[LONG]_CACHE,                  ;GET INDIRECT WORD
                                                  R(PRN)_[A+K[.4].RLOG,           ; WHILE UPDATING REGISTER
U 0394, 0018,0018,1180,40D8,0000,03C7             J/C.DF                          ;THEN JOIN COMMON CODE
```

```
                                         ;9013   ;GENERAL SPECIFIER (C-FORK) EVALUATION, CONTINUED
                                         ;9014
                                         ;9015   30C:     ;--------------------------------;
                                         ;9016            RC[T7]_LA.CTX,                   ;INDEX MODE, CONTEXT SHIFT INDEX
U 030C, 0060,C03D,0180,F9B8,0000,047E    ;9017            CALL,J7ASPC                      ; AND GO EVALUATE BASE OPERAND ADDRESS
                                         ;9018
                                         ;9019   36C:     ;--------------------------------;RETURN HERE WITH BASE OPERAND ADDRESS
                                         ;9020            D_Q,                             ;RESTORE DATA TO BE STORED
                                         ;9021            VA_D+LC,                         ;COMPUTE INDEXED ADDRESS
U 036C, 0C11,0814,0180,F800,0200,02D2    ;9022            DATA.TYPE?,J/C.M                 ;GET OR STORE NORMAL OR QUAD
                                         ;9023
                                         ;9024   30D:     ;--------------------------------;
                                         ;9025            VA_Q+LB.PC,                      ;D(R) DISPLACEMENT MODE.
                                         ;9026            Q_D,                             ;SAVE OLD D
                                         ;9027            CLR.IB.SPEC,                     ;DISCARD THE SPECIFIER
U 030D, D005,2814,01E0,F800,0200,02D2    ;9028            DATA.TYPE?,J/C.M                 ;GO GET THE OPERAND
                                         ;9029
                                         ;9030   30F:     ;--------------------------------;
                                         ;9031            Q_D,VA_Q+LB.PC,                  ;@D(R) DISPLACEMENT DEFERED
U 030F, D005,2014,01E0,F800,0200,03BE    ;9032            CLR.IB.SPEC                      ;DROP THE SPECIFIER
                                         ;9033
                                         ;9034   ; ******************************************
                                         ;9035   ; * Patch no. 055, PCS 030F trapped to WCS 117D *
                                         ;9036   ; ******************************************
                                         ;9037
                                         ;9038   FO.ABS.3BE:
                                         ;9039            ;--------------------------------;
U 03BE, 0000,003C,0180,4000,0000,03C7    ;9040            D[LONG]_CACHE                    ;GET INDIRECT, GO USE IT AS ADDR
                                         ;9041
                                         ;9042            ;--------------------------------;
                                         ;9043   C.DF:    VA_D,                            ;USE POINTER AS ADDRESS
                                         ;9044            D_Q,                             ;GET DATA TO STORE BACK TO D
U 03C7, 0C01,083C,0180,F800,0200,02D2    ;9045            DATA.TYPE?,J/C.M
```

```
                                          ;9046   ;HERE ARE VARIANTS OF THE C-FORK ENTRY POINTS FOR R=PC
                                          ;9047
                                          ;9048   314:     ;--------------------------------;
U 0314, 0000,003C,0180,F800,0000,0001     ;9049            J/RSVMOD                          ;PC REGISTER MODE
                                          ;9050
                                          ;9051   315:     ;--------------------------------;
U 0315, 0000,003C,0180,F800,0000,0001     ;9052            J/RSVMOD                          ;ILLEGAL REGISTER MODE, R=PC
                                          ;9053
                                          ;9054   316:     ;--------------------------------;
U 0316, 0000,003C,0180,F800,0000,0001     ;9055            J/RSVMOD                          ;PC QUAD REGISTER MODE
                                          ;9056
                                          ;9057   317:     ;--------------------------------;
U 0317, 0000,003C,0180,F800,0000,0001     ;9058            J/RSVMOD                          ;ILLEGAL QUAD REGISTER MODE, R=PC
                                          ;9059
                                          ;9060   318:     ;--------------------------------;
U 0318, 0000,003C,0180,F800,0000,0001     ;9061            J/RSVMOD                          ;(PC)
                                          ;9062
                                          ;9063   319:     ;--------------------------------;
                                          ;9064            D_Q,Q_D,                          ;(PC)+ IMMEDIATE
                                          ;9065            CLR.IB.SPEC,
U 0319, DC00,003E,01E0,F800,0000,0010     ;9066            RETURN10                          ;DONE
                                          ;9067
                                          ;9068   ; ************************************************
                                          ;9069   ; * Patch no. 047, PCS 0319 trapped to WCS 116F *
                                          ;9070   ; ************************************************
                                          ;9071
                                          ;9072   31A:     ;--------------------------------;
U 031A, 0000,003C,0180,F800,0000,0001     ;9073            J/RSVMOD                          ;-(PC)
                                          ;9074
                                          ;9075   31B:     ;--------------------------------;
                                          ;9076            VA_Q,Q_D,                         ;@(PC)+ ABSOLUTE MODE
                                          ;9077            CLR.IB.SPEC,
U 031B, D001,283C,01E0,F800,0200,02D2     ;9078            DATA.TYPE?,J/C.M
                                          ;9079
                                          ;9080   31C:     ;--------------------------------;
U 031C, 0000,003C,0180,F800,0000,0001     ;9081            J/RSVMOD                          ;INDEX MODE, R=PC
                                          ;9082
                                          ;9083   31D:     ;--------------------------------;
U 031D, 0000,003C,0180,F800,0000,0001     ;9084            J/RSVMOD                          ;NESTED INDEX MODE, R=PC
                                          ;9085
                                          ;9086   31F:     ;--------------------------------;
                                          ;9087            RC[T1]_Q,                         ;QUAD IMMEDIATE
                                          ;9088            Q_IB.DATA,                        ;GET SECOND PART INTO Q
                                          ;9089            CLR.IB.COND,
                                          ;9090            PC_PC+4,                          ;PUSH PC PAST SECOND PART
U 031F, F001,2B3C,01F0,F98E,0000,04D0     ;9091            IB.TEST?,J/C.IQ
                                          ;9092
                                          ;9093   ; ************************************************
                                          ;9094   ; * Patch no. 058, PCS 031F trapped to WCS 1184 *
                                          ;9095   ; ************************************************
```

```
                                              ;9096    ;SPECIAL CFORK STATES
                                              ;9097
                                              ;9098    387:     ;--------------------------------;HERE WE SHOULD NEVER GET
U 0387, 0000,003D,0180,F800,0000,0EE0         ;9099             CALL,J/EH.USEQ                   ;''UNUSED'' LOCATION FOUND IN IB ROM
                                              ;9100
                                              ;9101    ; ************************************************
                                              ;9102    ; * Patch no. 046, PCS 0387 trapped to WCS 116E *
                                              ;9103    ; ************************************************
                                              ;9104
                                              ;9105    37C:     ;--------------------------------;IB HAD NO DATA BECAUSE OF
U 037C, 0000,003D,0180,F800,0000,0E64         ;9106             CALL,J/IB.TBM                    ;TB MISS.  REFILL IT
                                              ;9107
                                              ;9108    37D:     ;--------------------------------;IB HAD NO DATA BECAUSE OF
U 037D, 0000,003D,0180,F800,0000,0B80         ;9109             CALL,J/IB.ERR                    ;ANY ERROR.  FIND OUT WHAT HAPPENED
                                              ;9110
                                              ;9111    37E:     ;--------------------------------;IB IS WAITING FOR DATA
                                              ;9112    SPEC:    LAB_R(SP1),                      ;STALL
                                              ;9113             Q_IB.DATA,
                                              ;9114             CLR.IB.COND,
                                              ;9115             PC_PC+N,
                                              ;9116             MCT/ALLOW.IB.READ,
U 037E, F000,003F,01F0,F847,0000,0300         ;9117             SUB/SPEC,J/C.FORK
                                              ;9118
                                              ;9119    37F:     ;--------------------------------;HERE IF INTERRUPT REQUEST UP
U 037F, 0000,003C,0180,F800,0000,04F8         ;9120             J/INT.B                          ;GO BACKUP REGISTERS, SERVE INTERRUPT
                                              ;9121
                                              ;9122    ;HERE TO WRITE BACK THE RESULT OF AN INSTRUCTION WITH A MODIFY DESTINATION.
                                              ;9123    ; ASSIGNED AN ADDRESS ON CFORK BECAUSE MANY 2-OPERAND INSTRUCTIONS ARE
                                              ;9124    ; EXECUTED BY THE SAME CODE AS THE 3-OPERAND COUNTERPART, AND CONCLUDE WITH
                                              ;9125    ; THE WRITE.DEST OPERATION, WHICH EVALUATES THE THIRD SPECIFIER IN THE
                                              ;9126    ; 3-OPERAND FORM, AND COMES HERE FOR THE 2-OPERAND FORM.
                                              ;9127
                                              ;9128    341:     ;--------------------------------;
                                              ;9129    STORE:   CACHE_D.INST.DEP,                ;STORE RESULT BY INSTR DATA TYPE
                                              ;9130             CLR.IB.OPC,                      ;MOVE NEXT INSTR INTO IB BYTE 0
U 0341, C000,C03C,0180,3004,4000,0062         ;9131             PC_PC+1,J/IRD                    ;DO NEXT INSTRUCTION
                                              ;9132
                                              ;9133    ;HERE IS THE SAME FUNCTION FOR QUAD/DOUBLE OPERATIONS
                                              ;9134
                                              ;9135    344:     ;--------------------------------;
                                              ;9136    STOR.Q:  VA_RC[T7],                       ;RELOAD OPERAND ADDRESS, WHICH GOT
U 0344, 0010,0038,0180,F938,0200,02D3         ;9137             J/C.WQ                           ; INCREMENTED IN FETCHING OPERAND
```

```
                                            ;HERE FOR THE SECOND AND SUBSEQUENT STATES OFF C-FORK SUBROUTINE

                                            =010        ;-------------------------------;GET HERE BY DATA.TYPE?
                                            C.M:        CACHE_D.INST.DEP,               ;STORE RESULT
U 02D2, C000,C03C,0180,3004,4000,0062                   CLR.IB.OPC,PC_PC+1,J/IRD        ;GO ON TO NEXT INSTR

                                                        ;-------------------------------;
U 02D3, 0000,C03C,0180,3000,0000,03F8       C.WQ:       CACHE_D.INST.DEP,J/C.WQ1        ;STORE QUAD/DBL RESULT,GO WRT 2nd PART

                                                        ;-------------------------------;
                                                        D_CACHE.INST.DEP,               ;GET MEMORY OPERAND
U 02D6, 0000,C03E,0180,5800,0000,0010                   RETURN10                        ;RETURN IT

                                                        ;-------------------------------;
U 02D7, 0000,C03C,0180,5800,0000,03D2                   D_CACHE.INST.DEP                ;GET FIRST PART OF QUAD OPERAND
                                            =;END OF DATA.TYPE TEST

                                                        ;-------------------------------;
                                                        RC[T2]_D,                       ;PUT LOW ADDR PART AWAY
                                                        ID_D.SYNC,                      ;SEND IT TO ACCELERATOR
U 03D2, 0001,007C,1580,BD93,0000,03E2                   VA_VA+4                         ;GET HIGH ADDR READY

                                                        ;-------------------------------;
U 03E2, 0000,003E,0180,4000,0000,0010                   D[LONG]_CACHE,RETURN10          ;GET HIGH ADDR PART

                                            ;HERE TO COMPLETE WRITE OF QUAD/DOUBLE OPERAND

                                                        ;-------------------------------;
                                            C.WQ1:      D_RC[T1],                       ;GET HIGH-ADDRESS DATA
U 03F8, 0810,0038,0180,F90B,0000,03FD                   VA_VA+4                         ;AND GO WRITE IT

                                                        ;-------------------------------;
                                            STOR.L:     CACHE_D[LONG],CLR.IB.OPC,       ;STORE SECOND PART OF QUAD RESULT
U 03FD, C000,003C,0180,3004,4000,0062                   PC_PC+1,J/IRD                   ;GO BACK TO IRD

                                            ;HERE FOR QUAD/DOUBLE IMMEDIATE OPERANDS

                                            =00         ;-------------------------------;
U 04D0, 0000,003D,0180,F800,0000,0E64       C.IQ:       CALL,J/IB.TBM

                                                        ;-------------------------------;
U 04D1, 0000,003D,0180,F800,0000,0B80                   CALL,J/IB.ERR

                                                        ;-------------------------------;
                                                        Q_IB.DATA,CLR.IB2-5,            ;
U 04D2, F000,0B3C,01F0,F800,0000,04D0                   IB.TEST?,J/C.IQ                 ;

                                                        ;-------------------------------;
U 04D3, DC00,003E,01E0,F800,0000,0010                   D_Q,Q_D,CLR.IB.SPEC,RETURN10    ;

                                            ; ********************************************
                                            ; * Patch no. 047, PCS 04D3 trapped to WCS 116F *
                                            ; ********************************************

                                            .LIST               ;Re-enable full listing
```

```
                        .TOC    "ARITH.MIC"
                        .TOC    "Revision 1.2"
                        ;        P. R. Guilbault

.NOBIN
.TOC    "       Revision History"

; 01    Remove absolute jumps.
;       Change macro names that deal with condition codes.
; 00    Delete MUL.MIC and put its code here.
;       Start of history

                        .BIN
                        .NOLIST         ;Disable listing of PCS code for quickie assemblies
```

```
;9207     .TOC    ''        Integer arithmetic     : Multiplication subroutine''
;9208
;9209     ;THE MULTIPLICATION IS DONE 2 BITS PER CYCLE.  THE MULTILICAND IS IN LB.
;9210     ;THE 2 TIMES MULTIPLICAND IS IN LC, AND THE MULTIPLIER IS IN D.
;9211     ;SC SHOULD HAVE 3/7/15, FOR B/W/L MULTIPLICATIONS.
;9212     ;THERE SHOULD BE ''ALU_0,D_D.RIGHT2,SI/ZERO,MUL?'' IN THE CALLING STATE.
;9213     ;DURING MULTIFYING, THE PARTIAL PRODUCT IS IN D.
;9214     ;WHEN DONE, THE PRODUCT IS IN Q AND D, WITH LSB'S IN D, AND SIGN EXTENDED IN Q.
;9215     ;
;9216     ;+VE MEANS LAST OPERATION IS POSITIVE, SUCH AS +2, OR +0.
;9217     ;-VE MEANS LAST OPERATION IS NEGATIVE, SUCH AS -1, OR -0.
;9218     ;+0 INDICATES TO DO DOUBLE RIGHT SHIFT BY 2 AND GO TO ''+VE'' COLUMNS FOR
;9219     ;NEXT OPERATION.  -2 INDICATES TO SUBTRACT 2 TIMES THE MULTIPLICAND
;9220     ;(LC HAS 2 TIMES MULTI'CAND, LB HAS MULTI'CAND) DO A DOUBLE RIGHT SHIFT
;9221     ;BY 2, AND GO TO ''-VE'' COLUMNS OF THE TABLE.
;9222     ;0XT, 1XT MEAN 0 EXTENDED, 1 EXTENDED WHEN SHIFTING, RESPECTIVELY.
;9223     ;RETURN TO RETURN ADDR .OR. 2 WHEN DONE FOR POSITIVE PRODUCT, SET SC TO -16.
;9224     ;RETURN TO RETURN ADDR .OR. 2 WHEN DONE FOR NEGATIVE PRODUCT, SET SC TO -16.
;9225     ;
;9226     ;                MULT'CAND IS POSITIVE             MULT'CAND IS NEGATIVE
;9227     ;       D<1:0>   +VE               -VE             +VE               -VE
;9228     ;                MULPP             MULPM           MULMP             MULMM
;9229     ;  ---------------------------------------------------------------------
;9230     ;       00       +0, 0XT           +1, 0XT         +0, 1XT (*)       +1, 1XT
;9231     ;       01       +1, 0XT           +2, 0XT         +1, 1XT           +2, 1XT
;9232     ;       10       -2, 1XT           -1, 1XT         -2, 0XT           -1, 0XT
;9233     ;       11       -1, 1XT           -0, 1XT         -1, 0XT           -0, 0XT
;9234
;9235     ; (*)   THIS IS ONLY TRUE ONCE A NON-ZERO BIT OF THE MULTIPLIER HAS BEEN
;9236     ;       ENCOUNTERED.  UNTIL THEN THE OPERATION USED IS  +0, 0XT
;9237     ;       (I.E., RECOGNIZING THE FACT THAT A NEGATIVE 0 IS POSITIVE)
```

```
                                          ;9238   ;       MULTIPLY LOOPS - EXPLANATION ON PREVIOUS PAGE
                                          ;9239
                                          ;9240   =000
U 0350, 0200,003E,6F00,F800,4084,6002     ;9241   MULPP:  SC_K[.FFF0],MULP.DONE,RETURN2           ;RETURN TO RETURN ADDR .OR. 2
U 0351, 0200,003E,6F00,F800,4084,6002     ;9242           SC_K[.FFF0],MULP.DONE,RETURN2           ;  FOR POS PRODUCT
U 0352, 0200,003E,6F00,F800,4084,6002     ;9243           SC_K[.FFF0],MULP.DONE,RETURN2           ;  (LAST EXTENDED BITS ARE 0S)
U 0353, 0200,003E,6F00,F800,4084,6002     ;9244           SC_K[.FFF0],MULP.DONE,RETURN2           ;  SET SC TO 16.
U 0354, 0281,2C3C,0740,F800,0084,A350     ;9245   MULPP.4:QD_QD.RIGHT2,      MUL.0XT,J/MULPP      ;+0, 0XT
U 0355, 028D,2C14,0740,F800,0084,A350     ;9246           QD_(Q+LB)D.RIGHT2,MUL.0XT,J/MULPP       ;+1, 0XT
U 0356, 0291,2C00,07C0,F800,0084,A3A0     ;9247           QD_(Q-LC)D.RIGHT2,MUL.1XT,J/MULPM       ;-2, 1XT
U 0357, 028D,2C00,07C0,F800,0084,A3A0     ;9248           QD_(Q-LB)D.RIGHT2,MUL.1XT,J/MULPM       ;-1, 1XT
                                          ;9249   =000
U 03A0, 0200,003E,6F80,F800,4084,6002     ;9250   MULPM:  SC_K[.FFF0],MULM.DONE,RETURN2           ;RETURN TO RETURN ADDR .OR. 2
U 03A1, 0200,003E,6F80,F800,4084,6002     ;9251           SC_K[.FFF0],MULM.DONE,RETURN2           ;  FOR NEG PRODUCT
U 03A2, 0200,003E,6F80,F800,4084,6002     ;9252           SC_K[.FFF0],MULM.DONE,RETURN2           ;  (LAST EXTENDED BITS ARE 1S)
U 03A3, 0200,003E,6F80,F800,4084,6002     ;9253           SC_K[.FFF0],MULM.DONE,RETURN2           ;  SET SC TO 16.
U 03A4, 028D,2C14,0740,F800,0084,A350     ;9254           QD_(Q+LB)D.RIGHT2,MUL.0XT,J/MULPP       ;+1, 0XT
U 03A5, 0291,2C14,0740,F800,0084,A350     ;9255           QD_(Q+LC)D.RIGHT2,MUL.0XT,J/MULPP       ;+2, 0XT
U 03A6, 028D,2C00,07C0,F800,0084,A3A0     ;9256           QD_(Q-LB)D.RIGHT2,MUL.1XT,J/MULPM       ;-1, 1XT
U 03A7, 0281,2C3C,07C0,F800,0084,A3A0     ;9257           QD_QD.RIGHT2,      MUL.1XT,J/MULPM      ;-0, 1XT
                                          ;9258
                                          ;9259
                                          ;9260
                                          ;9261   =000
U 03B0, 0200,003E,6F80,F800,4084,6002     ;9262   MULMP:  SC_K[.FFF0],MULM.DONE,RETURN2           ;RETURN TO RETURN ADDR .OR. 2
U 03B1, 0200,003E,6F80,F800,4084,6002     ;9263           SC_K[.FFF0],MULM.DONE,RETURN2           ;  FOR NEG PRODUCT
U 03B2, 0200,003E,6F80,F800,4084,6002     ;9264           SC_K[.FFF0],MULM.DONE,RETURN2           ;  (LAST EXTENDED BITS ARE 1S)
U 03B3, 0200,003E,6F80,F800,4084,6002     ;9265           SC_K[.FFF0],MULM.DONE,RETURN2           ;  SET SC TO 16.
U 03B4, 0281,2C3C,07C0,F800,0084,A3B0     ;9266           QD_QD.RIGHT2,      MUL.1XT,J/MULMP      ;+0, 1XT
U 03B5, 028D,2C14,07C0,F800,0084,A3B0     ;9267           QD_(Q+LB)D.RIGHT2,MUL.1XT,J/MULMP       ;+1, 1XT
U 03B6, 0291,2C00,0740,F800,0084,A3F0     ;9268           QD_(Q-LC)D.RIGHT2,MUL.0XT,J/MULMM       ;-2, 0XT
U 03B7, 028D,2C00,0740,F800,0084,A3F0     ;9269           QD_(Q-LB)D.RIGHT2,MUL.0XT,J/MULMM       ;-1, 0XT
                                          ;9270
                                          ;9271   =000
U 03F0, 0200,003E,6F00,F800,4084,6002     ;9272   MULMM:  SC_K[.FFF0],MULP.DONE,RETURN2           ;RETURN TO RETURN ADDR .OR. 2
U 03F1, 0200,003E,6F00,F800,4084,6002     ;9273           SC_K[.FFF0],MULP.DONE,RETURN2           ;  FOR POS PRODUCT
U 03F2, 0200,003E,6F00,F800,4084,6002     ;9274           SC_K[.FFF0],MULP.DONE,RETURN2           ;  (LAST EXTENDED BITS ARE 0S)
U 03F3, 0200,003E,6F00,F800,4084,6002     ;9275           SC_K[.FFF0],MULP.DONE,RETURN2           ;  SET SC TO 16.
U 03F4, 028D,2C14,07C0,F800,0084,A3B0     ;9276           QD_(Q+LB)D.RIGHT2,MUL.1XT,J/MULMP       ;+1, 1XT
U 03F5, 0291,2C14,07C0,F800,0084,A3B0     ;9277           QD_(Q+LC)D.RIGHT2,MUL.1XT,J/MULMP       ;+2, 1XT
U 03F6, 028D,2C00,0740,F800,0084,A3F0     ;9278           QD_(Q-LB)D.RIGHT2,MUL.0XT,J/MULMM       ;-1, 0XT
U 03F7, 0281,2C3C,0740,F800,0084,A3F0     ;9279           QD_QD.RIGHT2,      MUL.0XT,J/MULMM      ;-0, 0XT
                                          ;9280
                                          ;9281
                                          ;9282   =100                                            ;NEGATIVE MULTIPLIES START HERE
U 0294, 0281,2C3C,0740,F800,0084,A294     ;9283   MULMZ:  QD_QD.RIGHT2,      MUL.0XT,J/MULMZ      ;+0, 0XT
U 0295, 028D,2C14,07C0,F800,0084,A3B0     ;9284           QD_(Q+LB)D.RIGHT2,MUL.1XT,J/MULMP       ;+1, 1XT
U 0296, 0291,2C00,0740,F800,0084,A3F0     ;9285           QD_(Q-LC)D.RIGHT2,MUL.0XT,J/MULMM       ;-2, 0XT
U 0297, 028D,2C00,0740,F800,0084,A3F0     ;9286           QD_(Q-LB)D.RIGHT2,MUL.0XT,J/MULMM       ;-1, 0XT
```

```
                                        ;9287   .TOC    "       Integer arithmetic      : Divide subroutine"
                                        ;9288
                                        ;9289   ;DESCRIPTION:
                                        ;9290   ;
                                        ;9291   ;       RESTORING DIVIDE SUBROUTINE
                                        ;9292   ;
                                        ;9293   ;       ENTER AT DIVOX TO PRODUCE POSITIVE QUOTIENT,
                                        ;9294   ;       ENTER AT DIV1X TO PRODUCE NEGATIVE QUOTIENT.
                                        ;9295   ;
                                        ;9296   ;INPUTS:
                                        ;9297   ;       HIGH DIVIDEND IN D, LOW DIVIDEND IN Q
                                        ;9298   ;       DIVISOR IN LB, STEP COUNT IN SC.
                                        ;9299   ;       ALL NUMBERS CONSIDERED POSITIVE
                                        ;9300   ;
                                        ;9301   ;OUTPUTS:
                                        ;9302   ;       QUOTIENT (+ OR -) IN Q, REMAINDER IN D. SC = 0.
                                        ;9303   ;
                                        ;9304   ;RETURNS:       ALWAYS AT 8
                                        ;9305
                                        ;9306   =011
                                        ;9307   DIV00:  ;011---------------------------;
                                        ;9308           K[.8000],Q_D.RIGHT,            ;
                                        ;9309           SI/ZERO,D_Q,                   ;
U 01C3, 0C41,003E,45C0,F800,4000,0008   ;9310           INTRPT.STROBE,RETURN8          ;
                                        ;9311
                                        ;9312   DIVOX:  ;111---------------------------;
                                        ;9313           DK/DIV,Q_Q.LEFT,               ;
                                        ;9314           SHF/LEFT,SI/DIV,               ;
                                        ;9315           SC_SC-K[.1],ALU_D-LB,          ;
U 01C7, 042D,0C00,06A8,F800,0084,A1C3   ;9316           MUL?,J/DIV00                   ;+/+
                                        ;9317
                                        ;9318   DIV111: ;------------------------------;
                                        ;9319           K[.8000],Q_0-Q,                ;
U 040E, 001F,0002,45C0,F800,4000,0008   ;9320           INTRPT.STROBE,RETURN8          ;
                                        ;9321   =011
                                        ;9322   DIV11:  ;011---------------------------;
                                        ;9323           Q_D.RIGHT,                     ;
U 01D3, 0C41,003C,01C0,F800,0000,040E   ;9324           SI/ZERO,D_Q,J/DIV111           ;-/+: -RMD, -QUOT
                                        ;9325
                                        ;9326   DIV1X:  ;111---------------------------;
                                        ;9327           DK/DIV,Q_Q.LEFT,               ;
                                        ;9328           SHF/LEFT,SI/DIV,               ;
                                        ;9329           SC_SC-K[.1],ALU_D-LB,          ;
U 01D7, 042D,0C00,06A8,F800,0084,A1D3   ;9330           MUL?,J/DIV11                   ;-/+
                                              ;9331   .TOC ''  Integer arithmetic     : MULB2, MULB3, MULW2, MULW3, MULL2, MULL3''
                                              ;9332
                                              ;9333   :         MUL(B/W/L)2      MULR.RX, PROD.MX
                                              ;9334   :         MUL(B/W/L)3      MULR.RX, MULD.RX, PROD.WX
                                              ;9335   ;INTEGER MUL'S, ENTER HERE FROM B-FORK
                                              ;9336   ;MUL B/W/L 2: * -- REG
                                              ;9337   ;THE OPERANDS ARE IN LA AND D REGISTERS.
                                              ;9338
                                              ;9339   22D:      ;------------------------;
U 022D, 0800,003C,01E0,F800,0000,0411         ;9340             D_LA, Q_D                 :   NEED TO SIGN EXT M'IER
                                              ;9341
                                              ;9342             ;------------------------;
                                              ;9343             D_D.SXT[INST.DEP],        :   SIGN EXT M'IER
                                              ;9344             RC[T1]_ALU,               :   SAVE IN RC 1
U 0411, 0802,C03C,4980,F988,0084,60A4         ;9345             SC_K[.FF]                 :   SETUP TO GET CONSTANT 100 (HEX)
                                              ;9346
                                              ;9347   =01*0     ;00----------------------;   (IR0 = 0)
                                              ;9348             Q_Q.SXT[INST.DEP],        :   SIGN EXTEND MUL'CAND
                                              ;9349             ID[T0]_D,                 :   SAVE MUL'IER
                                              ;9350             SC_SC+T,                  :   SC GETS 100 (HEX)
U 00A4, 0002,E03D,C1C0,3C00,0080,C430         ;9351             CALL,J/MUL.S              :
                                              ;9352
                                              ;9353             ;01----------------------;   POS
                                              ;9354             ALU_Q, SET.CC(INST),      :   SET PSL<V> IF OVERFLOW
U 00A5, 0001,E03C,0180,F800,0070,00AC         ;9355             J/MUL.0                   :
                                              ;9356
                                              ;9357   MUL.0:    ;10----------------------;   RETURN HERE FOR PROD = 0
                                              ;9358             R(PRN)_D,                 :   STORE RESULT 0
                                              ;9359             DT/INST.DEP,              :   WRITE B/W/L TO R
                                              ;9360             CLR.IB.OPC,PC_PC+1,       :   UPDATE IB, PC
U 00AC, C001,C03C,0180,F8DC,4000,0062         ;9361             J/IRD                     :   GOTO NEXT INSTR
                                              ;9362
                                              ;9363             ;11----------------------;   NEG
                                              ;9364             ALU_Q+K[.1],              :   SET PSL<V> IF OVERFLOW
                                              ;9365             SET.CC(INST),             :
U 00AD, 0019,E014,0580,F800,0070,00AC         ;9366             J/MUL.0                   :
                                              ;9367   =;END
```

```
                                             ;INTEGER MUL'S, ENTER HERE FROM C-FORK
                                             ;MUL B/W/L 2: * -- NOT REG, * -- * -- *
                                             ;THE OPERANDS ARE IN D AND Q REGISTERS.

                                             381:
                                             MUL:      ;---------------------------;
                                                       D_D.SXT[INST.DEP],           ;   SIGN EXT M'IER
                                                       RC[T1]_ALU,                  ;   SAVE IN RC 1
U 0381, 0802,C03C,4980,F988,0084,6506                  SC_K[.FF]                    ;   SETUP TO GET CONSTANT 100 (HEX)

                                             =0110     ;00-------------------------;
                                                       Q_Q.SXT[INST.DEP],           ;   SIGN EXTEND MUL'CAND
                                                       ID[T0]_D,                    ;   SAVE MUL'IER
U 0506, 0002,E03D,C1C0,3C00,0080,C430                  SC_SC+T, CALL, J/MUL.S       ;   SC GETS 100 (HEX), CALL MUL SUBR

                                                       ;01-------------------------;   POS
                                                       ALU_Q,SET.CC(INST),          ;   SET PSL<V> IF OVERFLOW
U 0507, F001,E03F,01F0,F847,0070,0300                  WRITE.DEST,J/WRD             ;   WRITE RESULT

                                                       ;10-------------------------;   RETURN HERE FOR PROD = 0
                                                       D_K[ZERO],                   ;   PROD IS 0
                                                       SET.CC(INST),                ;   SET COND CODES
U 050E, F818,C03B,19FC,F847,0070,0300                  WRITE.DEST,J/WRD             ;   WRITE RESULT

                                                       ;11-------------------------;   NEG
                                                       ALU_Q+K[.1],                 ;   SET PSL<V> IF OVERFLOW
                                                       SET.CC(INST),                ;
U 050F, F019,E017,05F0,F847,0070,0300                  WRITE.DEST,J/WRD             ;   WRITE RESULT
                                             =;END
```

```
                                                  ;       COMMON SIGNED MULTIPLY SUBROUTINE FOR BYTE, WORD, LONGWORD

                                                  ;INPUTS:
                                                  ;       SIGN-EXTENDED MULTIPLIER IN D, COPIES IN RC[T1] AND ID[T0].
                                                  ;       MULTIPLICAND (ALSO SIGN EXTENDED) IN Q
                                                  ;       SC = 100(HEX)
                                                  ;       INSTRUCTION DECODE ROMS DETERMINE DATA TYPE

                                                  ;OUTPUTS:
                                                  ;       D = LOW 32 BITS OF PRODUCT
                                                  ;       Q = BITS OF PRODUCT WHICH DON'T FIT IN RESULT

                                                  ;RETURNS:
                                                  ;       RETURNS AT 1 IF PRODUCT > 0
                                                  ;       RETURNS AT 8 IF PRODUCT = 0
                                                  ;       RETURNS AT 9 IF PRODUCT < 0

                                                  ;TEMPORARIES:
                                                  ;       R15     USED TO SAVE MULTIPLICAND
                                                  ;       STATE   USED TO HOLD DATA-TYPE DEPENDENT SHIFT COUNTS
                                                  ;       FE      DITTO
                                                  ;       LA,LB,LC USED IN MULTIPLY LOOP


                                                  MUL.S:  ;---------------------------------------;
                                                          R[R15]_Q, D_Q,                  ;       SAVE M'CAND
                                                          SC_SC+K[SC],                    ;       SC NOW CONTAINS 200
U 0430, 0C01,2D3C,1D80,FAF8,0084,8135                     D.NE.0?                         ;       MUL'IER IS 0?

                                                  =101    ;0--------------------------------------;
                                                          D_K[ZERO],N&Z_ALU.V&C_0,        ;       PROD IS 0 SINCE MUL'IER IS 0
U 0135, 0818,003A,1980,F800,0050,0008                     RETURN8                         ;       WRITE RESULT 0

                                                          ;1--------------------------------------;
                                                          Q_K[SC].CTX,                    ;       SET SHF COUNT FOR B,W,L
U 0137, 0078,C038,1DC0,FA78,0000,043A                     LAB_R[R15]                      ;       LATCH MUL'CAND
                                                  =;END
                                                          ;---------------------------------------;
                                                          SC_Q(EXP), STATE_Q(EXP),        ;       SC GETS COUNT (4,8,16) FOR B,W,L VIA EBMX
                                                          FE_Q(EXP), Q_D, DK/SHF,         ;       SAVE CT TO REMEMBER B,W,L
U 043A, 082D,2038,01E0,F980,1588,6068                     RC[T0]_LB.LEFT.SI/ZERO          ;       RC 0 GETS 2 TIMES M'CAND

                                                  =0*     ;0--------------------------------------;
                                                          D_RC[T1],Q_0,                   ;       D GETS M'IER
                                                          STATE_STATE+FE,                 ;       STATE HAS # BITS (8,16,32) FOR B,W,L
U 0068, 0810,0D39,01F8,F908,1400,82A1                     CALL, SIGNS?, J/MUL.6           ;       POS OR NEG MUL'CAND?
```

```
                                                       ;1----------------------;
                                                       ALU_D, CLK.UBCC,        :  D31 HAS HI BIT OF PROD - ALU.N=SIGN
                                                       STATE_STATE-K[.20],     :  STATE GETS -24, -16, 0 FOR B,W,L
U 006A, 0001,003C,758C,F800,1494,A464                  SC_EALU                 :  MORE IMPORTANTLY, SO DOES SC
                                         =;END
                                                       ;-----------------------;
U 0464, 0D00,003C,0180,F800,0000,046D                  D_DAL.SC                :  ALIGN PRODUCT FOR BYTE, WORD, LONG INSTR

                                                       ;-----------------------;
                                                       ALU_D, N&Z_ALU.V&C_0,   :  SET COND CODES, N & Z
                                                       DT/INST.DEP,            :  DATA TYPE SET FOR B/W/L
U 046D, 0001,DB3E,0180,F800,0050,0001                  ALU.N?, RETURN1         :  IS PROD POS OR NEG?

                                         =001
                                         MUL.6:        ;00---------------------:  M'CAND IS 0
                                                       D_K[ZERO], Q_0,         :  PROD IS 0 SINCE MUL'CAND IS 0
                                                       N&Z_ALU.V&C_0,          :  SET COND CODES
U 02A1, 0818,003A,19F8,F800,0050,0002                  RETURN2                 :  WRITE RESULT 0

                                                       ;01---------------------;
                                                       SC_SC-K[.1],            :  SC HAS LOOP COUNT (3,7,15) FOR B,W,L
                                                       LC_RC[T0], ALU_0(A),    :  LATCH 2 TIMES MUL'CAND, SETUP ALU[1:0]
                                                       D_D.RIGHT2, SI/ZERO,    :  SHIFT MUL'IER BY 2 BITS
U 02A3, 0203,0C3C,0580,F900,0084,A350                  MUL?, J/MULPP           :  GOTO MULT ROUTINE

                                                       ;10---------------------;  M'CAND IS MOST NEG NUMBER FOR MULL
                                                       Q_0-D,D_0,              :  NEG M'IER
U 02A5, 0F1F,2000,01C0,F800,0000,0475                  J/MUL.8                 :

                                                       ;11---------------------;
                                                       SC_SC-K[.1],            :  SC HAS LOOP COUNT (3,7,15) FOR B,W,L
                                                       LC_RC[T0], ALU_0(A),    :  LATCH 2 TIMES MUL'CAND, SETUP ALU[1:0]
                                                       D_D.RIGHT2, SI/ZERO,    :  SHIFT MUL'IER BY 2 BITS
U 02A7, 0203,0C3C,0580,F900,0084,A294                  MUL?, J/MULMZ           :  GOTO MULT ROUTINE AT INITIAL NEG ENTRY PT
                                         =;END

                                         MUL.8:        ;-----------------------;
                                                       ALU_Q,Q_ALU.RIGHT,      :  ARITH SHF RIGHT <Q,D>
                                                       D_D.RIGHT,SI/ASHR,      :
U 0475, 0641,203E,00C0,F800,0000,0002                  RETURN2                 :
```

```
                                        .TOC    ""      Integer arithmetic      : EMUL"

                                        ;       EMUL    7A      MULR.RL, MULD.RL, ADD.RL, PROD.WQ
                                        ;EMUL:  EXTEND INTEGER MULTIPLICATION.
                                        ;THE MULT'CAND IS IN D, AND THE MULT'IER IS IN Q.

                                        ;       DOES NOT USE THE SAME ROUTINE USED BY NORMAL INTEGER MULTIPLIES,
                                        ;       BUT INSTEAD CALLS THE LOW LEVEL MUL ROUTINE DIRECTLY.
                                        ;       THE ADDEND IS ADDED AS A SEPARATE STEP INSTEAD OF "FOR FREE"
                                        ;       DURING THE MULTIPLY LOOP BECAUSE OF OVERFLOW PROBLEMS IN THE LOOP.

                                        389:
                                        EMUL:   ;---------------------------;
                                                R[R15]_D,                   ;       R15 GETS MULT'CAND
                                                Q_D, D_Q,                   ;       SWAP D, AND Q
U 0389, 0C01,0D3C,01EC,FAF8,0000,01A5           D.NE.0?                     ;       MULT'CAND .NE. 0?

                                        =101    ;0--------------------------;       NO:  MULT'CAND = 0, THEREFORE PRODUCT = 0
                                                D_0, ID[T0]_D,              ;       SET PROD TO 0
U 01A5, 0F00,003C,C180,3C00,0000,02A2           J/EMUL.2                    ;       GOTO ADD ADDEND

                                                ;1--------------------------;       YES: MULT'CAND NE 0, CHECK IF MULT'IER = 0
                                                RC[T0]_Q.LEFT, SI/ZERO,     ;       RC0 GETS 2 TIMES MULT'CAND
U 01A7, 0021,2D3C,0180,F980,0000,02B1           SIGNS?                      ;       MULT'IER .NE. 0?
                                        =;END

                                        =001    ;00-------------------------;       NO:  MULT'IER = 0, THEREFORE PROD = 0
                                                Q_0,                        ;       SET PROD TO 0
U 02B1, 0000,003C,01F8,F800,0000,02A2           J/EMUL.2                    ;       GOTO ADD ADDEND

                                                ;01-------------------------;       YES: PROD .NE. 0, M'CAND POS
                                                LAB_R[R15],                 ;       LB GETS MULT'CAND
                                                Q_0, SC_K[.F],              ;       PARTIAL PROD RESET TO 0, LOOP COUNT SET FOR 16.
U 02B3, 0000,003C,61F8,FA78,0084,62A0           J/EMUL.T                    ;       GOTO POS ROUTINE

                                                ;10-------------------------;       NO:  MULT'IER = 0, THEREFORE PROD = 0
                                                Q_0,                        ;       SET PROD TO 0
U 02B5, 0000,003C,01F8,F800,0000,02A2           J/EMUL.2                    ;       GOTO ADD ADDEND

                                                ;11-------------------------;       YES: PROD .NE. 0, M'CAND NEG
                                                LAB_R[R15],                 ;       LB GETS MULT'CNAD
U 02B7, 0000,003C,61F8,FA78,0084,61C0           Q_0, SC_K[.F]               ;       PARTIAL PROD RESET TO 0, LOOP COUNT SET FOR 16.

                                        ; ***********************************************
                                        ; * Patch no. 029, PCS 02B7 trapped to WCS 1161 *
                                        ; ***********************************************
                                        =;END

                                        =0**0*
                                        AR.PA.29:
                                                ;0--------------------------;
                                                LC_RC[T0],                  ;       LATCH 2 TIMES M'CAND IN LC
                                                ALU_0(A),                   ;       SET ALU[1:0] TO 0
                                                D_D.RIGHT2, SI/ZERO,        ;       READY FOR MULT ROUTINE
U 01C0, 0203,0C3D,0180,F900,0000,0294           CALL, MUL?, J/MULMZ         ;       GOTO NEG M'CAND MULTIPLICATION ROUTINE
```

```
                                        ;9539
                                        ;9540   =0**1*   ;1----------------------;
                                        ;9541            R[R15] Q,               :  SAVE PROD <H>
                                        ;9542            CALL,INTERRUPT.REQ?,    :
U 01C2, 0001,2E3D,0180,FAF8,0000,037E   ;9543            J/SPEC                  :  GET ADDEND
                                        ;9544
                                        ;9545   =1**1*   ;1**1*------------------;
                                        ;9546            D_D+Q, CLK.UBCC,        :  ADD ADDEND, CLOCK IN CARRY
                                        ;9547            LAB R[R15],             :  LATCH PROD <H>
U 01D2, 081D,0D14,0180,FA78,0010,0156   ;9548            D31?, J/EMUL.3          :  IS ADDEND NEG?
                                        ;9549   =;END
```

```
                                          ;9550   =0**0*
                                          ;9551   EMUL.1: ;----------------------;
                                          ;9552           LC_RC[T0],             ;  LATCH 2 TIMES M'CAND IN LC
                                          ;9553           ALU_0(A),              ;  SET ALU[1:0] TO 0
                                          ;9554           D_D.RIGHT2, SI/ZERO,   ;  READY FOR MULT ROUTINE
U 02A0, 0203,0C3D,0180,F900,0000,035?    ;9555           CALL, MUL?, J/MULPP    ;  GOTO POS M'CAND MULTIPLICATION ROUTINE
                                          ;9556   =0**1*
                                          ;9557   EMUL.2: ;----------------------;
                                          ;9558           R[R15]_Q,              ;  SAVE PROD <H>
                                          ;9559           CALL,INTERRUPT.REQ?,   ;
U 02A2, 0001,2E3D,0180,FAF8,0000,037E    ;9560           J/SPEC                 ;  GET ADDEND
                                          ;9561
                                          ;9562   =1**1*  ;1**1*-----------------;
                                          ;9563           D_D+Q, CLK.UBCC,       ;  ADD ADDEND, CLOCK IN CARRY
                                          ;9564           LAB_R[R15],            ;  LATCH PROD <H>
U 02B2, 081D,0D14,0180,FA78,0010,0156    ;9565           D31?                   ;  IS ADDEND NEG?
                                          ;9566   =;END
                                          ;9567
                                          ;9568   =110
                                          ;9569   EMUL.3: ;0---------------------;
                                          ;9570           ALU_D,SET.CC(INST),    ;  SET COND CODES PART 1
U 0156, 0001,C33C,0180,F800,0070,0250    ;9571           C31?, J/EMUL.4         ;  CARRY TO PROD <H>?
                                          ;9572
                                          ;9573           ;1---------------------;
                                          ;9574           ALU_D,SET.CC(INST),    ;  SET COND CODES PART 1
U 0157, 0001,C33C,0180,F800,0070,00E8    ;9575           C31?                   ;  CARRY TO PROD <H>?
                                          ;9576   =;END
                                          ;9577
                                          ;9578   =0*     ;0---------------------;
                                          ;9579           Q_LA-K[.1],RC[T1]_ALU, ;  GET PROD <H>
U 00E8, 0018,0000,05C0,F988,0000,0492    ;9580           J/EMUL.6               ;  GOTO DECREMENT IT BY 1
                                          ;9581
                                          ;9582           ;1---------------------;
                                          ;9583           RC[T1]_LA,             ;  PROD <H>
                                          ;9584           N_AMX.Z_TST,           ;  SET COND CODES PART 2
U 00EA, 0000,003C,0180,F988,0030,0603    ;9585           J/WRDST                ;  GOTO WRITE DEST
                                          ;9586   =;END
                                          ;9587
                                          ;9588   EMUL.6: ;----------------------;
                                          ;9589           ALU_Q,N_AMX.Z_TST,     ;  SET COND CODES PART 2
U 0492, F001,203F,01F0,F847,0030,0300    ;9590           WRITE.DEST, J/WRD      ;  WRITE RESULT
                                          ;9591
                                          ;9592   =0*     ;0---------------------;
                                          ;9593   EMUL.4: RC[T1]_LA,             ;  PROD <H>
                                          ;9594           N_AMX.Z_TST,           ;  SET COND CODES PART 2
U 0250, 0000,003C,0180,F988,0030,0603    ;9595           J/WRDST                ;  GOTO WRITE DEST
                                          ;9596
                                          ;9597           ;1---------------------;
                                          ;9598           Q_LA+K[.1],RC[T1]_ALU, ;  RC1 GETS PROD <H>
U 0252, 0018,0014,05C0,F988,0000,0492    ;9599           J/EMUL.6               ;  GOTO SET COND CODES
                                          ;9600   =;END
```

```
                                                  ;9601    .TOC "  Integer arithmetic    : DIVB2, DIVB3, DIVW2, DIVW3, DIVL2, DIVL3"
                                                  ;9602
                                                  ;9603    ;        DIV(B/W/L)2     DIVR.RX, QUO.MX
                                                  ;9604    ;        DIV(B/W/L)3     DIVR.RX, DIVD.RX, QUO.WX
                                                  ;9605    ;INTERGER DIV, ENTER AT B.FORK WITH D'END IN LA AND D'SOR IN D.
                                                  ;9606
                                                  ;9607    22C:     ;-----------------------;
U 022C, 0800,003C,01E0,F800,0000,04B1             ;9608             Q_D, D_LA               ;   D GETS D'END, Q GETS D'SOR, SIGN EXT LATTER
                                                  ;9609
                                                  ;9610             ;-----------------------;
                                                  ;9611             D_Q.SXT[INST.DEP],      ;   SIGN EXT D'SOR
                                                  ;9612             Q_D,                    ;   Q GETS D'END
U 04B1, 0802,E03C,19E0,F800,1404,64C3             ;9613             STATE_K[ZERO]           ;   CLEAR FLAG (USED FOR NEG QUOT)
                                                  ;9614
                                                  ;9615             ;-----------------------;
U 04C3, 0001,003C,0180,FAF8,0000,0150             ;9616             R[R15]_D                ;   SAVE D'SOR
                                                  ;9617
                                                  ;9618    =0**0    ;00---------------------;
                                                  ;9619             D_Q.SXT[INST.DEP], Q_0, ;   SXT EXT D'END, Q=0 TO HACK CONSTRAINT @DIV.2
                                                  ;9620             LAB_R[R15],             ;   LATCH D'SOR
U 0150, 0802,ED3D,01F8,FA78,0000,0374             ;9621             CALL[SIGNS?,J/DIV.S     ;   D.NE.0? D31?
                                                  ;9622
                                                  ;9623             ;01---------------------;   RETURN HERE FOR DIV BY 0
U 0151, 0000,003C,0180,F800,0000,0280             ;9624             J/DIV.ZO                ;
                                                  ;9625
                                                  ;9626             ;10---------------------;   RETURN HERE WITH D HAS QUOT
                                                  ;9627             Q_D.SXT[INST.DEP]. D_0, ;   Q GETS QUOT, D=0 FOR CONSTRAINT HACK & NEGATE
U 0158, 0F02,D73C,01C0,F800,0000,0142             ;9628             STATE0?                 ;   HAVE TO NEGATE QUOTIENT?
                                                  ;9629    =;END
                                                  ;9630
                                                  ;9631    =**10    ;0----------------------;   NO: +/+ OR -/-
                                                  ;9632             R(PRN)_Q, N&Z_ALU.V&C_0,;   SET COND CODES
                                                  ;9633             DT/INST.DEP,            ;   WRITE ONLY B/W/L IN R
U 0142, 0001,ED3C,0180,F8D8,0050,0180             ;9634             Q31?,J/DIV.OV           ;   OVERFLOW?
                                                  ;9635
                                                  ;9636             ;1----------------------;   YES: +/- OR -/+
                                                  ;9637             R(PRN)_ALU,ALU_D-Q,     ;   NEGATE QUOTIENT
                                                  ;9638             N&Z_ALU.V&C_0,          ;   SET COND CODES
                                                  ;9639             DT/INST.DEP,            ;   WRITE ONLY B/W/L IN R
                                                  ;9640             CLR.IB.OPC,PC_PC+1,     ;   UPDATE IB, PC
U 0143, C01D,C000,0180,F8DC,4050,0062             ;9641             J/IRD                   ;   GOTO NEXT INST
                                                  ;9642    =;END
                                                  ;9643
                                                  ;9644    =0*0     ;0----------------------;   NO OVERFLOW
                                                  ;9645    DIV.OV:  CLR.IB.OPC, PC_PC+1,    ;
U 0180, C000,003C,0180,F804,4000,0062             ;9646             J/IRD                   ;
                                                  ;9647
                                                  ;9648    =1*0     ;1----------------------;   OVERFLOW - MOST NEG NUMBER / -1
U 0184, 0000,003D,31F0,2C00,0000,0DFC             ;9649             Q_ID[CES], CALL[INOVFL] ;   GO SET V AND TRAP CODE
                                                  ;9650
                                                  ;9651             ;-----------------------;   RETURN FROM INOVFL HERE
                                                  ;9652             CLR.IB.OPC,PC_PC+1,     ;   UPDATE IB, PC
U 0185, C000,003C,0180,F804,4000,0062             ;9653             J/IRD                   ;   NEXT INST
                                                  ;9654    =;END
                                                  ;9655
```

```
                                          ;9656   =0
                                          ;9657   DIV.Z0: ;0---------------------;
                                          ;9658           Q_ID[CES],              ;  GET CES
U 0280, 0000,003D,31F0,2C00,0000,0DFD     ;9659           CALL,J/INDIV0           ;  CALL SETUP CES
                                          ;9660
                                          ;9661           ;1---------------------;
                                          ;9662           CLR.IB.OPC,PC_PC+1,     ;  UPDATE IB, PC
U 0281, C000,003C,0180,F804,4000,0062     ;9663           J/IRD                   ;  NEXT INST
                                          ;9664   =;END
```

```
                                         ;INTERGER DIV, ENTER AT C.FORK WITH D'END IN D AND D'SOR IN Q.
                                         380:
                                         DIV:      ;-----------------------;
                                                   ALU_Q.SXT[INST.DEP]+K[ZERO],     ;  SIGN EXT D'SOR
                                                   D_ALU,CLK.UBCC,Q_D,      ;  Q GETS D'END
U 0380, 081A,E014,19E0,F800,1414,64EA              STATE_K[ZERO]            ;  CLEAR FLAG (USED FOR NEG QUOT)

                                                   ;-----------------------;
U 04EA, 0001,003C,0180,FAF8,0000,0270              R[R15]_D                 ;  SAVE D'SOR

                                         =0**0     ;00---------------------;
                                                   D_Q.SXT[INST.DEP], Q_0,  ;  SXT EXT D'END, Q=0 FOR CONSTRAINT HACK @DIV.2
                                                   LAB_R[R15],              ;  LATCH D'SOR
U 0270, 0802,ED3D,01F8,FA78,0000,0374              CALL,SIGNS?,J/DIV.S      ;  D.NE.0? D31?

                                                   ;01---------------------;  RETURN HERE FOR DIV BY 0
                                                   D_Q.SXT[INST.DEP],       ;  SXT EXT D'END
U 0271, 0802,FB3C,0180,F800,0000,0334              IR0?, J/DIV.Z            ;  IF 2-OPR INST, DO NOT CHANGE QUOT OPR

                                                   ;10---------------------;  RETURN HERE WITH D HAS QUOT
                                                   Q_D.SXT[INST.DEP], D_0,  ;  Q GETS QUOT, D=0 FOR CONSTRAINT HACK
U 0278, 0F02,D73C,01C0,F800,0000,022A              STATE3-0?                ;  HAVE TO NEGATE QUOT?
                                         =;END

                                         =**10     ;0----------------------;  NO: +/+ OR -/-
                                                   ALU_Q,D_ALU,DT/INST.DEP, ;  MOVE QUOT TO D
                                                   N&Z_ALU.V&C_0,           ;  SET COND CODES
U 022A, 0801,ED3C,0180,F80C,0050,02E0              Q31?, J/DIV.OV3          ;  OVERFLOW?

                                                   ;1----------------------;  YES: +/- OR -/+
                                                   D_D-Q, N&Z_ALU.V&C_0,    ;  SET COND CODES BY NEG QUOT
                                                   DT/INST.DEP,             ;  DATA TYPE SET FOR B/W/L
U 022B, F81D,C003,01F0,F847,0050,0300              WRITE.DEST               ;  WRITE RESULT
                                         =;END

                                         =0*0      ;0----------------------;  NO OVERFLOW
U 02E0, F000,003F,01F0,F847,0000,0300    DIV.OV3:  WRITE.DEST               ;  JUST WRITE & LEAVE

                                         =1*0      ;1----------------------;  OVERFLOW - MAX NEG # / -1
U 02E4, 0000,003D,31F0,2C00,0000,0DFC              Q_ID[CES], CALL[INOVFL]  ;  SET TRAP CODE AND V BIT

                                                   ;-----------------------;  INOVFL RETURNS HERE
U 02E5, F000,003F,01F0,F847,0000,0300              WRITE.DEST               ;  WRITE RESULT
                                         =;END

                                         =*100                              ; (SINCE DIVISOR=0, ALU.N=0)
                                         DIV.Z:    ;00---------------------;  2-OPR INST, DO NOT CHANGE QUOT OPR
U 0334, 0000,003C,0180,F800,0000,0280              J/DIV.ZO                 ;

                                         =*110     ;10---------------------;  3-OPR INST, QUOT OPR GETS D'END
U 0336, 0000,003D,31F0,2C00,0000,0DFD              Q_ID[CES], CALL[INDIVO]  ;  GET CES, GO SET TRAP CODE & V BIT

                                                   ;11---------------------;
U 0337, F000,003F,01F0,F847,0000,0300              WRITE.DEST               ;  WRITE QUOT
                                         =;END
```

```
                                        ;9720   ;SUBROUTINE TO DO BYTE, WORD OR LONGWORD DIVISION.
                                        ;9721   ;USES RESTORING DIVIDE SUBROUTINE DIVOX.
                                        ;9722   ;ENTER AT DIV.S WITH BEN/SIGNS TESTING DIVISOR-WHICH WAS IN D TILL CALLING STATE
                                        ;9723
                                        ;9724   ;INPUTS:  DIVISOR (SIGN EXTENDED TO LONGWORD) IN R15 WITH A COPY IN LA&LB.
                                        ;9725   ;         DIVIDEND (ALSO SIGN EXTENDED) IN D
                                        ;9726   ;         Q = STATE = 0.
                                        ;9727
                                        ;9728   ;OUTPUTS: Q = ABSOLUTE VALUE OF QUOTIENT
                                        ;9729   ;         D = ABSOLUTE VALUE OF REMAINDER
                                        ;9730   ;         STATE<0> = DESIRED QUOTIENT SIGN(1 MEANS QUOTIENT IN Q NEEDS NEGATING)
                                        ;9731
                                        ;9732   ;RETURNS: RETURNS AT 1 IF DIVISOR = 0
                                        ;9733   ;         RETURNS AT 8 OTHERWISE
                                        ;9734
                                        ;9735   ;TEMPORARIES: SC        USED FOR STEP COUNTER
                                        ;9736   ;             R15,LA,LB DESTROYED
                                        ;9737
                                        ;9738   =100
                                        ;9739   DIV.S:  ;00---------------------; D'SOR IS 0
                                        ;9740           ALU_D, N&Z_ALU.V&C_0,   ; SET COND CODES N & Z
                                        ;9741           DT/INST.DEP,            ; DATA TYPE SET FOR B/W/L
U 0374, 0001,C03E,0180,F800,0050,0001   ;9742           RETURN1                 ; D'SOR IS 0
                                        ;9743
                                        ;9744   ; ***********************************************
                                        ;9745   ; * Patch no. 034, PCS 0374 trapped to WCS 1166 *
                                        ;9746   ; ***********************************************
                                        ;9747
                                        ;9748   =110    ;10---------------------; D'SOR .NE. 0, AND IS POS
                                        ;9749   DIV.0:  Q_K[.8].CTX,            ; SET LOOP CT OF 8,16,32 FOR B,W,L
                                        ;9750           LAB_R[R15],             ; LATCH ABS(D'SOR)
U 0376, 0078,CD38,01C0,FA78,0000,02C2   ;9751           D31?,J/DIV.2            ; IS D'END POS OR NEG?
                                        ;9752
                                        ;9753           ;11---------------------; D'SOR .NE. 0, AND IS NEG
                                        ;9754           R[R15]_0-LB,            ; R15 GETS ABS(D'SOR)
U 0377, 000F,0000,0180,FAF8,1400,0376   ;9755           STATE_STATE+1,J/DIV.0   ; EXCLUSIVE OR STATE[00] AS FLAG FOR NEGATE QUOT
                                        ;9756   =;END
                                        ;9757   =*10
                                        ;9758   DIV.2:  ;0----------------------; TO ALIGN D'END, LEFT JUSTIFIED IN Q
                                        ;9759           ALU_0-Q,SC_ALU,         ; SC GETS -8,-16.,-32. FOR B,W,L
                                        ;9760           Q_D, D_0,               ; Q GETS D'END
U 02C2, 0F1F,0000,01E0,F800,0082,04EE   ;9761           J/DIV.3                 ; GOTO ALIGN D'END
                                        ;9762
                                        ;9763           ;1----------------------; D GETS ABS(D'END) FOR NEG D'END
                                        ;9764           D_0-D,                  ;
U 02C3, 081F,2000,0180,F800,1400,02C2   ;9765           STATE_STATE+1,J/DIV.2   ; EXCLUSIVE OR STATE[00] AS FLAG FOR NEGATE QUOT
                                        ;9766   =;END
                                        ;9767   DIV.3:  ;-----------------------;
                                        ;9768           D_DAL.SC,               ; D GETS D'END LEFT JUSTIFIED
U 04EE, 0D1B,0000,1D80,F800,0082,04F1   ;9769           SC_0-K[SC]              ; SC GETS LOOP CT 8,16.,32. FOR B,W,L
                                        ;9770
                                        ;9771           ;-----------------------;
                                        ;9772           Q_D, D_0,               ; Q GETS D'END
U 04F1, 0F00,003C,01E0,F800,0000,01C7   ;9773           J/DIVOX                 ; GOTO DIVIDE ROUTINE
```

```
                                          .TOC    ""      Integer arithmetic    : EDIV"

                                          ;       EDIV    (7B)    DIVR.RL, DIVD.RQ, QUO.WL, REM.WL
                                          ;INTERGER EXTENDED DIVIDE, EDIV, WITH D'END IN <D, RC1> AND D'SOR IN Q.
                                          388:
                                          EDIV:   ;-----------------------;
                                                  R[R15]_Q,               ; R15 GETS D'SOR
                                                  D_Q, Q_D,               ; D GETS D'SOR, Q GETS D'END <H>
U 0388, 0C01,203C,C1E0,3EF8,0000,0501             ID[T0]_D                ; SAVE D'END<H> IN CASE WE OVERFLOW

                                                  ;-----------------------;
                                                  SC_K[.20].ALU,          ; SET LOOP COUNT
                                                  STATE_0(A), LC_RC[T1],  ; LATCH D'END <L>
                                                  D_Q, Q_D,               ; D GETS D'END <H>, Q GETS D'SOR
U 0501, 0C1B,CD38,75E0,F908,148A,6584             SIGNS?                  ; D'SOR = 0? POS OR NEG?

                                          =0100   ;00---------------------; D'SOR = 0
                                                  D_RC[T1],               ; D GETS DIVIDEND<31:0>
U 0584, 0810,0038,0180,F908,0050,0364             N&Z_ALU.V&C_0, J/EDIV.Z ; SET CCL 5 ON D'END <L>
                                          =0110
                                          EDIV.1: ;10---------------------; D'SOR IS POS
                                                  LAB_R[R15],             ; LATCH D'SOR IN LB
                                                  ALU_LC, CLK.UBCC,       ; SET ALU.Z IF DIVIDEND<L>=0
U 0586, 0010,0D39,0180,FA78,0010,0256             CALL, D31?, J/EDIV.6    ; D'END POS OR NEG?

                                                  ;11---------------------; D'SOR IS NEG
                                                  R[R15]_0-Q,             ; R15 GETS ABS(D'SOR)
U 0587, 001F,0000,0180,FAF8,1400,C586             STATE_STATE+1,J/EDIV.1  ; EXCL OR STATE[00] AS FLAG FOR NEGATE QUOT

                                          =1110   ;-----------------------;
                                                  ALU_0+D, CLK.UBCC,      ; RETURN HERE WITH QUOT IN D, REM IN Q
U 058E, 001F,3714,0180,F800,0010,0412             STATE0?                 ; SHOULD QUOTIENT BE POS OR NEG?
                                          =;END
                                          =**10   ;0----------------------; QUOT IS POS
                                                  ALU_D, N&Z_ALU.V&C_0,   ; SET CONDITION CODES ON QUOTIENT
U 0412, 0001,1B3C,0180,F800,0050,00E6             ALU.N?, J/EDIV.9        ; CHECK FOR OVERFLOW AND GO STORE

                                                  ;1----------------------; QUOT IS NEG
U 0413, 0019,0100,0580,F800,0010,0310             ALU_D-K[.1],CLK.UBCC, Z?: SET UP OVFLO TEST & CHECK IF 0

                                          ; **************************************************
                                          ; * Patch no. 011, PCS 0413 trapped to WCS 114B *
                                          ; **************************************************
                                          =;END
                                          =0      ;0----------------------; QUOT < 0
                                                  D_0-D, N&Z_ALU.V&C_0,   ; NEGATE QUOTIENT
U 0310, 081F,3B00,0180,F800,0050,00E6             ALU.N?, J/EDIV.9        ; GO CHECK OVERFLOW (POS NUM > 2**31)

                                                  ;1----------------------; QUOT = 0
                                                  ID[T0]_D, D_Q, SC_0(A), ; QUO = D, REM = Q.
                                                  ALU_0(A), N&Z_ALU.V&C_0,; SET COND CODES FOR A 0 RESULT
                                                  INTRPT.STROBE,          ; USE EMODF CODE TO STORE RESULTS
U 0311, 0C03,003C,C180,3C00,54D8,708C             STATE_0(A), J/EMODF.11  ; SINCE A PAIR OF LWORDS IS A PAIR OF LWORDS
```

```
                                        ;9827   =0
                                        ;9828   EDIV.2: ;0----------------------;   OVERFLOW COMES HERE
                                        ;9829           Q_ID[CES],              ;   GET CES
U 0320, 0000,003D,31F0,2C00,0000,0DFC   ;9830           CALL,J/INOVFL           ;   CALL SETUP CES
                                        ;9831
                                        ;9832           ;1----------------------;
                                        ;9833           ID[T0]_D, D_0, SC_0(A), ;   QUO = D'END<L>, REM = 0,
                                        ;9834           INTRPT.STROBE,          ;   USE EMODF CODE TO STORE RESULTS
U 0321, 0F03,003C,C180,3C00,5488,708C   ;9835           STATE_0(A), J/EMODF.11  ;   SINCE A PAIR OF LONGWORDS IS A PAIR OF LONGWORDS
                                        ;9836   =;END
                                        ;9837   EDIV.3: ;-----------------------;
U 0541, 0010,0D38,01C0,F908,0000,0162   ;9838           Q_RC[T1],Q31?           ;   Q GETS D'END <L>, OVERFLOW?
                                        ;9839
                                        ;9840   =01*    ;0----------------------;   OK (D KNOWN +)
U 0162, 0000,003C,0180,F800,0000,01C7   ;9841           J/DIVOX                 ;   CALL DIV SUBRT
                                        ;9842
                                        ;9843           ;1----------------------;   OVERFLOW
                                        ;9844           D_RC[T1],               ;   Q GETS D'END <L>
U 0166, 0810,0038,0180,F908,0050,0320   ;9845           NEZ_ALU.V&C_0, J/EDIV.2 ;   SET CCL 5 ON D'END <L>
                                        ;9846   =;END
                                        ;9847   =110    ;0----------------------;   D'END POS
                                        ;9848   EDIV.6: ALU_LA-D-1, Q_ALU,      ;   CHK FOR OVERFLOW
U 0256, 001C,2008,01C0,F800,0000,0541   ;9849           J/EDIV.3                ;   CALL DIV SUBROUTINE
                                        ;9850
                                        ;9851           ;1----------------------;   D'END NEG
                                        ;9852           STATE_STATE+1,          ;   EXCL OR STATE[00] AS FLAG FOR NEGATE QUOT
U 0257, 0013,0100,0180,F990,1400,0358   ;9853           ALU_0-LC, RC[T2]_ALU, Z?;   RC[T2] GETS ABS(D'END <L>), SKIP IF ZERO
                                        ;9854   =;END
                                        ;9855
                                        ;9856   =0      ;0----------------------;   D'END <L> NOT ZERO
                                        ;9857           D_NOT.D,                ;   D'END <H> IS 1'S COMP TO NEGATE D'END
U 0358, 0801,0028,0180,F800,0000,058A   ;9858           J/EDIV.7                ;   GOTO DIV SUBRT
                                        ;9859
                                        ;9860           ;1----------------------;   D'END IS ZERO
U 0359, 081F,2000,0180,F800,0000,058A   ;9861           D_0-D                   ;   NEG D'END <H> INSTEAD 1'S COMP OF IT
                                        ;9862   =;END
                                        ;9863   EDIV.7: ;-----------------------;
                                        ;9864           ALU_LA-D-1, Q_ALU,      ;   CHK FOR OVERFLOW
U 058A, 0F1C,2008,C1C0,3C00,0000,058C   ;9865           ID[T0]_D, D_0           ;   SAVE D AND CLR IT FOR CONSTRAINT HACK
                                        ;9866
                                        ;9867           ;-----------------------;
U 058C, 0810,0D38,C1F0,2D10,0000,0051   ;9868           D_RC[T2], Q_ID[T0], Q31?;   D = D'END<L>, Q = D'END<H>, OVERFLOW?
                                        ;9869
                                        ;9870   =0**    ;0----------------------;   OK
U 0051, 0C00,003C,01E0,F800,0000,01D7   ;9871           D_Q, Q_D, J/DIV1X       ;   CALL DIV SUBRT
                                        ;9872
                                        ;9873           ;1----------------------;   OVERFLOW
                                        ;9874           D_RC[T1],               ;   Q GETS D'END <L>
U 0055, 0810,0038,0180,F908,0050,0320   ;9875           NEZ_ALU.V&C_0, J/EDIV.2 ;   SET CCL 5 ON D'END <L>
                                        ;9876   =;END
```

```
                                      ;9877                                          ; CONSTRAINT FOR ALU.N AND PSL.N BRANCHES
                                      ;9878   =011*    ;0---------------------;     EXIT CODE - NO OVERFLOW
                                      ;9879   EDIV.9: ID[T0]_D, D_Q, SC_0(A), ;     QUO = D, REM = Q,
                                      ;9880            INTRPT.STROBE,         ;     USE EMODF CODE TO STORE RESULTS
U 00E6, 0C03,003C,C180,3C00,5488,708C ;9881            STATE_0(A), J/EMODF.11 ;     SINCE A PAIR OF LWORDS IS A PAIR OF LWORDS
                                      ;9882
                                      ;9883            ;1---------------------;     OVERFLOW
                                      ;9884            D_RC[T1], N&Z_ALU.V&C_0,;    QUOTIENT = DIVIDEND<31:0>
U 00EE, 0810,0038,0180,F908,0050,0320 ;9885            J/EDIV.2
                                      ;9886   =0
                                      ;9887   EDIV.Z: ;0---------------------;     DIVIDE BY 0 COMES HERE
                                      ;9888            Q_ID[CES],             ;     GET C.E.S.
U 0364, 0000,003D,31F0,2C00,0000,0DFD ;9889            CALL, J/INDIV0         ;     GO STICK DIV BY 0 CODE IN CES
                                      ;9890
                                      ;9891            ;1---------------------;
                                      ;9892            ID[T0]_D, D_0, SC_0(A), ;     QUO = D'END<L>, REM = 0,
                                      ;9893            INTRPT.STROBE,         ;     USE EMODF CODE TO STORE RESULTS
U 0365, 0F03,003C,C180,3C00,5488,708C ;9894            STATE_0(A), J/EMODF.11 ;     SINCE A PAIR OF LWORDS IS A PAIR OF LWORDS
                                      ;9895
                                      ;9896   .LIST           ;Re-enable full listing
```

```
                                        ;9897   .TOC    "INDEX.MIC"
                                        ;9898   .TOC    "Revision 1.0"
                                        ;9899   ;        P. R. Guilbault
                                        ;9900
;9901   .NOBIN
;9902   .TOC    ""      Revision History"
;9903
;9904   ; 01    Change macro names that deal with condition codes.
;9905   ; 00    Start of history
;9906
                                        ;9907   .BIN
                                        ;9908   .NOLIST         ;Disable listing of PCS code for quickie assemblies
```

```
                                     .TOC    ""      Index instruction      : INDEX"

                                     ;       opcode(0A)      subscript.rl, low.rl, high.rl, size.rl, indexin.rl,
                                     ;                       indexout.wl

                                     ;ALGORITHM:

                                     ;       The operation specified for this instruction is:

                                     ;               indexin <- {indexin + subscript}*size;
                                     ;               if {subscript LSS low} or {subscript GTR high}
                                     ;               then {subscript range trap}

                                     ;       On entry to this routine from C-FORK, the indexin and subscript
                                     ;       operands have been fetched by A-FORK and B-FORK routines.  The flow
                                     ;       is as follows:

                                     ;       1) Get 'high' limit operand per SPEC subroutine
                                     ;       2) If 'subscript' operand less than 'low' operand, setup
                                     ;          subscript range trap
                                     ;       3) Get 'size' operand per SPEC subroutine
                                     ;       4) If 'subscript' operand less than 'high' operand, setup
                                     ;          subscript range trap
                                     ;       5) Get 'indexin' operand per SPEC subroutine
                                     ;       6) Calculate (subscript + indexin) and set condition codes
                                     ;          for next step
                                     ;       7) If 'size' operand = 1, write above result per WRITE.DEST function
                                     ;       8) Calculate (subscript + indexin)*size using MUL.S subroutine
                                     ;       9) Set condition codes and write result per WRITE.DEST function


                                     ;STORAGE/REGISTER ALLOCATION:

                                     ;       D-REG   'low' operand on entry
                                     ;       Q-REG   'subscript' operand on entry
                                     ;       ID[CES] Range trap code
                                     ;       ID[T0]  temporary
                                     ;       RC[T1]  'size' for MUL.S routine
                                     ;       RC[T2]  temporary



                                     3C9:    ;-----------------------;
U 03C9, 001D,E000,0180,F800,0070,000E        ALU_Q-D,  SET.CC(INST)  ;  TEST FOR 'LOW' LEQ 'SUBSCRIPT'

                                     =0**1*  ;CALL CONSTRAINT BLOCK FOR SPEC ROUTINE

                                             ;0**1*------------------;
                                             D_Q,                    ;  MOVE SUBSCRIPT TO D-REG
U 000E, 0C00,003D,0180,F800,0000,037E        CALL,   J/SPEC          ;  GO GET 'HIGH' OPERAND

                                             ;1**1*------------------;  RETURN FROM SPEC ROUTINE
U 001E, 0000,1A3C,0180,F800,0000,0066        PSL.N?                  ;  WAS SUBSCRIPT LESS THAN LOW LIMIT?
```

```
                                        ;9962    =0011*  ;CALL CONSTRAINT BLOCK FOR SPEC AND PSL.N BRANCH(BEN/PSL)
                                        ;9963
                                        ;9964            ;0011*-------------------;   *** SUBSCRIPT RANGE TRAP DETECTED ***
                                        ;9965            Q_ID[CES],               ;   GET CP ERR/STATUS REG
                                        ;9966            D_Q,                     ;   SAVE Q-REG IN D-REG
                                        ;9967            RC[T2]_D,                ;   SAVE D-REG IN RC STACK
U 0066, 0C01,003D,31F0,2D90,0000,0591   ;9968            CALL,  J/INDEX.9         ;   GO SET EXCEPTION TRAP CODE & RETURN8
                                        ;9969
                                        ;9970            ;0111*-------------------;   *** LOW LIMIT OK, GO GET SIZE OPERAND ***
                                        ;9971            ALU_D-Q,  SET.CC(INST),  ;   TEST FOR 'HIGH' > 'SUBSCRIPT'
                                        ;9972            D_Q,                     ;   MOVE SUBSCRIPT TO D-REG
U 006E, 0C1D,C001,0180,F800,0070,037E   ;9973            CALL,  J/SPEC            ;   GO GET SIZE (SUBSCRIPT/Q-REG ON RETURN)
                                        ;9974
                                        ;9975    =1111*  ;1111*-------------------;   *** RETURN10/12 FROM SPEC ***
                                        ;9976            ALU_D-K[.1], CLK.UBCC,   ;   TEST 'SIZE' FOR EQ 1
U 007E, 0019,1A00,0580,F800,0010,01A6   ;9977            PSL.N?                   ;   WAS 'HIGH' > 'SUBSCRIPT'
                                        ;9978
                                        ;9979    =0011*  ;CALL CONSTRAINT BLOCK FOR SPEC AND PSL.N BRANCH(BEN/PSL)
                                        ;9980
                                        ;9981            ;0011*-------------------;   *** SUBSCRIPT RANGE TRAP DETECTED ***
                                        ;9982            Q_ID[CES],               ;   GET CP ERR/STATUS REG
                                        ;9983            D_Q,                     ;   SAVE Q-REG IN D-REG
                                        ;9984            RC[T2]_D,                ;   SAVE D-REG IN RC STACK
U 01A6, 0C01,003D,31F0,2D90,0000,0591   ;9985            CALL,  J/INDEX.9         ;   GO SET EXCEPTION TRAP CODE & RETURN8
                                        ;9986
                                        ;9987            ;0111*-------------------;   *** HIGH LIMIT OK, GO GET INDEX OPERAND ***
                                        ;9988            ID[T0]_D,                ;   SAVE MULTIPLIER(SIZE)
                                        ;9989            RC[T1]_D,                ;   SET UP T1 FOR MULTIPLY ROUTINE
                                        ;9990            D_Q,                     ;   MOVE SUBSCRIPT TO D-REG
U 01AE, 0C01,003D,C180,3D88,0000,037E   ;9991            CALL,  J/SPEC            ;   GO GET INDEXIN (SUBSCRIPT/Q-REG ON RETURN)
                                        ;9992
                                        ;9993    =1111*  ;1111*-------------------;   *** RETURN10/12 FROM SPEC ***
                                        ;9994            D&Q_D+Q, N&Z_ALU.V&C_0,  ;   ADD SUBSCRIPT TO INDEXIN
                                        ;9995            SC_K[.FF],               ;   START SETUP OF 100
U 01BE, 081D,0114,49C0,F800,00D4,6390   ;9996            Z?                       ;   WAS SIZE EQ TO 1?
                                        ;9997
                                        ;9998
                                        ;9999
                                        ;10000   =0      ;ALU CC<Z> EQ 0?(BEN/Z)
                                        ;10001
                                        ;10002           ;0----------------------;   *** SIZE OPERAND NOT EQ 1 ***
                                        ;10003           SC_SC+1,                ;   100 ->SC
U 0390, 0000,003C,0180,F800,0080,C516   ;10004           J/INDEX.3               ;
                                        ;10005
                                        ;10006   ; ***********************************************
                                        ;10007   ; * Patch no. 061, PCS 0390 trapped to WCS 1187 *
                                        ;10008   ; ***********************************************
                                        ;10009
                                        ;10010           ;1----------------------;   *** SIZE OPERAND EQ 1 ***
U 0391, F000,003F,01F0,F847,0000,0300   ;10011           WRITE.DEST              ;   WRITE RESULT BY GOING TO C-FORK aWRD:
```

```
                                         =0110      ;CALL CONSTRAINT BLOCK FOR MUL.S ROUTINE

                                         INDEX.3:;0110-  ----------------;
U 0516, 0000,003D,0180,F800,0000,0430              CALL,    J/MUL.S           ;   GO DO (INDEXIN+SUBSCRIPT)*SIZE

                                                   ;0111-------------------;   RETURN FROM MUL.S
                                                   ALU_D,  N&Z_ALU.V&C_0,  ;   SET CONDITION CODES FOR RESULT
U 0517, F001,003F,01F0,F847,0050,0300              WRITE.DEST              ;   WRITE RESULT BY GOING TO C-FORK @ WRD:

                                                   ;1110-------------------;   RETURN FROM MUL.S
                                                   ALU_D,  N&Z_ALU.V&C_0,  ;   SET CONDITION CODES FOR RESULT
U 051E, F001,003F,01F0,F847,0050,0300              WRITE.DEST              ;   WRITE RESULT BY GOING TO C-FORK @ WRD:

                                                   ;1111-------------------;   RETURN FROM MUL.S
                                                   ALU_D,  N&Z_ALU.V&C_0,  ;   SET CONDITION CODES FOR RESULT
U 051F, F001,003F,01F0,F847,0050,0300              WRITE.DEST              ;   WRITE RESULT BY GOING TO C-FORK @ WRD:

                                         INDEX.9:;-----------------------;
U 0591, 0019,2030,79C0,F800,0000,05C2              Q_Q.OR.K[.30]           ;

                                                   ;-----------------------;
                                                   Q_D,                    ;   RESTORE Q-REG
U 05C2, 0819,2030,31E0,F800,0000,05C5              D_Q.OR.K[.40]           ;

                                                   ;-----------------------;
                                                   D_RC[T2],               ;   RESTORE D-REG
                                                   ID[CES]_D,              ;   SET SUBSCRIPT RANGE TRAP FLAG
U 05C5, 0810,003A,3180,3D10,0000,0008              RETURN8                 ;   RETURN AND CONTINUE INSTRUCTION

                                         .LIST             ;Re-enable full listing
```

```
                                        ;10042  .TOC     "FLOAT.MIC"
                                        ;10043  .TOC     "Revision 2.14"
                                        ;10044  ;         P. R. Guilbault
                                        ;10045
;10046  .NOBIN
;10047  .TOC    ""      Revision History"
;10048
;10049  ; 02    Fix POLY FPD problem that backs up the regs on interrupt
;10050  ;       Add general WCS region
;10051  ;       Convert EMODF to floating faults
;10052  ;       Convert POLYF/D to floating faults
;10053  ;       Fix (MUL,DIV)F2 destination register when floating fault.
;10054  ;       Fix POLYF when argument or partial product is zero.
;10055  ;       Remove absolute jumps.
;10056  ;       Add CVTRDL.G tag for G&h
;10057  ;       Change macro names that deal with conditions codes.
;10058  ; 01    Delete FLOATW.MIC and put code here. Use .REGION to get it into WCS.
;10059  ;        FLOATW 00      Create this file by merging MULD.MIC, EMOD.MIC, POLY.MIC
;10060  ;        FLOATW         Remove macros that were defined MULD.MIC and put in MACRO.MIC
;10061  ;        FLOATW         Start of history
;10062  ;       Add LIST to enable listing of WCS code for WCS only listing
;10063  ; 00    Delete DBL.MIC, CVT2F.MIC, CVTFI2.MIC, ACBFD2.MIC and put code here.
;10064  ;       Start of history
;10065
                                        ;10066  .BIN
                                        ;10067  .NOLIST         ;Disable listing of PCS code for quickie assemblies
                                        ;10068  .TOC    "       F & D floating point   : CMPF"
                                        ;10069
                                        ;10070  ;THE SPECIFIER 1 OPERAND IS COMPARED WITH THE SPECIFIER 2 OPERAND.
                                        ;10071  ;PSL<N> <-- SP1 LSS SP2
                                        ;10072  ;PSL<Z> <-- SP1 EQL SP2
                                        ;10073  ;PSL<V> <-- 0
                                        ;10074  ;PSL<C> <-- 0
                                        ;10075  ;
                                        ;10076  ;ENTER HERE FROM DP2.WITH D CONTAINS DST. Q CONTAINS SRC.
                                        ;10077  ;
                                        ;10078  ;       THE COMPARISON IS DONE IN A SIGN-INDEPENDENT MANNER; THE
                                        ;10079  ;       CONDITION CODES ARE SET FROM THE SOURCE OR FROM THE
                                        ;10080  ;       NEGATED DESTINATION, WHICHEVER IS HIGHER IN MAGNITUDE.
                                        ;10081  ;       IF THE SIGNS ARE DIFFERENT. THE TWO TESTS ARE THE SAME (OF COURSE)
                                        ;10082  ;       SO NO MAGNITUDE COMPARISON IS DONE.
                                        ;10083  ;
                                        ;10084  ;       MAGNITUDES ARE COMPARED BY COMARING THE EXPONENTS, AND THEN
                                        ;10085  ;       COMPARING THE FRACTIONS IF THE EXPS ARE EQUAL.
                                        ;10086  ;       THE ONLY SPECIAL CASE IS THAT BOTH EXPONENTS = 0 MEANS EQUALITY.
                                        ;10087
                                        ;10088  3C1:
                                        ;10089  CMPF:   ;-------------------------------;
                                        ;10090          ALU_Q(B),                       ;GET SRC AND DEST EXPONENTS
                                        ;10091          SC_ALU(EXP).FE_D(EXP),          ;
                                        ;10092          RC[T0]_ALU,                     ;SAVE AND UNPACK SRC
                                        ;10093          Q_ALU(FRAC).SS_ALU15,           ;
U 03C1, 001D,0038,01C9,F980,099B,65D4   ;10094          CHK.FLT.OPR, CLK.UBCC           ;CHECK FOR -0 AND SET CC'S ON EXPS
                                        ;10095
                                        ;10096          ;-------------------------------;
                                        ;10097          R[R15]_D,                       ;SAVE DST, UNPACK DST, GET EXP DIFF
                                        ;10098          D_D(FRAC).EALU_SC-FE,           ;
                                        ;10099          SS_SS.XOR.ALU15&SD_ALU15,       ;REMEMBER IF SIGNS DIFFERNET
U 05D4, 0901,123C,0185,FAF8,0810,A4E9   ;10100          CHK.FLT.OPR.CLK.UBCC.EALU?      ;SET CC ON EXP DIFF, TEST EXPS=0
                                        ;10101
                                        ;10102  =1001   ;1001---------------------------;
                                        ;10103          ALU_R[R15].XOR.K[.8000],        ;SET CONDITION CODES FROM -SRC2
                                        ;10104          SET.CC(INST),                   ;
U 04E9, C018,C020,4580,FA7C,4070,0062   ;10105          CLR.IB.OPC.PC_PC+1.J/IRD        ;
                                        ;10106
                                        ;10107          ;1011---------------------------;
                                        ;10108          ALU_Q-D.CLK.UBCC,               ;EXPS<>0 - CMP FRACS, TST EXP DIFF
U 04EB, 001D,3200,0180,F800,0010,0592   ;10109          EALU?.J/CKDIF0                  ;
                                        ;10110
                                        ;10111          ;1101---------------------------;
                                        ;10112          ALU_K[ZERO].SET.CC(INST),       ;DST, SRC = 0
U 04ED, C018,C038,1980,F804,4070,0062   ;10113          CLR.IB.OPC.PC_PC+1.J/IRD        ;
                                        ;10114
                                        ;10115          ;1111---------------------------;
                                        ;10116          ALU_RC[T0], SET.CC(INST),       ;DST = 0, SRC .NE. 0
U 04EF, C010,C038,0180,F904,4070,0062   ;10117          CLR.IB.OPC.PC_PC+1.J/IRD        ;SET CC'S FROM SRC
```

```
                                          ;10118  =0010
                                          ;10119  CKDIF0: ;0010---------------------------;
                                          ;10120          ALU_RC[T0],SET.CC(INST),        ;
U 0592, C010,C038,0180,F904,4070,0062     ;10121          CLR.IB.OPC,PC_PC+1,J/IRD        ;SRC > DST
                                          ;10122
                                          ;10123          ;0011---------------------------;
                                          ;10124          ALU_RC[T0],SET.CC(INST),        ;
U 0593, C010,C038,0180,F904,4070,0062     ;10125          CLR.IB.OPC,PC_PC+1,J/IRD        ;SRC > DST
                                          ;10126
                                          ;10127          ;0110---------------------------;
U 0596, 0000,1B3C,0180,F800,0000,065A     ;10128          ALU?, J/CHECKF                  ;SRC(EXP)=DST(EXP) - TEST FRAC DIFF
                                          ;10129
                                          ;10130          ;0111---------------------------;
                                          ;10131          ALU_RC[T0],SET.CC(INST),        ;DIFF SIGNS: CC SET AS SRC
U 0597, C010,C038,0180,F904,4070,0062     ;10132          CLR.IB.OPC,PC_PC+1,J/IRD        ;
                                          ;10133
                                          ;10134          ;1010---------------------------;
                                          ;10135          ALU_R[R15].XOR.K[.8000],        ;DST > SRC
                                          ;10136          SET.CC(INST),                   ;
U 059A, C018,C020,4580,FA7C,4070,0062     ;10137          CLR.IB.OPC,PC_PC+1,J/IRD        ;
                                          ;10138
                                          ;10139          ;1011---------------------------;
                                          ;10140          ALU_R[R15].XOR.K[.8000],        ;DST > SRC
                                          ;10141          SET.CC(INST),                   ;
U 059B, C018,C020,4580,FA7C,4070,0062     ;10142          CLR.IB.OPC,PC_PC+1,J/IRD        ;
                                          ;10143  =
                                      .TOC    "       F & D floating point  : ADDF, SUBF"

                                      ;ADDF2/SUBF2    Short literal, Register
                                      ;HERE FROM IRD, WITH LA CONTAINS SP2 (DST), AND Q CONTAINS SP1 (SRC).
                                      040:
                                              ;---------------------------------;
                                              ALU_Q,                             ;SETUP EXP
                                              D_Q,SC_Q(EXP),SS_ALU15,            ;
U 0040, 0C01,203C,0181,F800,0888,6044         CHK.FLT.OPR                        ;


                                      ;ADDF2/SUBF2    Register, Register
                                      ;       HERE FROM IRD, WITH LA CONTAINS SP2 (DST), AND D CONTAINS SP1 (SRC).
                                      044:
                                      ADDF:   ;0****0100----------------------;
                                              Q_D,RC[T0]_LA,                     ;SAVE DST OPERAND
                                              D_LA(FRAC),SC_NABS(SC-FE),         ;UNPACK DST FP, GET EXP DIFFERENCE
                                              SGN/ADD.SUB,                       ;SS_+/- INDICATOR, SET SD
                                              CHK.FLT.OPR,CLK.UBCC,              ;RSV OPD FAULT IF -0, SET ALUS CC
U 0044, 0900,123D,01E6,F980,0890,E4F9         CALL,EALU?,J/ADDFX                 ;CHECK FOR 0 EXPS

                                      144:    ;1****0100----------------------;RET FOR RESULT=0
                                              R(SP1)_K[ZERO],                    ;
                                              EALU_K[ZERO],SET.CC(INST),         ;
U 0144, 4018,C038,1980,F8C5,4074,6062         CLR.IB0-1,PC_PC+2,J/IRD            ;

                                      14C:
                                      ADDFDN: ;1****1100----------------------;RET FOR ADDF2/SUBF2
                                              EALU_SC,R(SP1)_PACK.FP,            ;PACK RESULT
                                              SET.CC(INST),                      ;SET COND CODES
                                              CLR.IB0-1,PC_PC+2,                 ;UPDATE PC, POP IB
U 014C, 4008,D438,0180,F8C5,4070,05D1         SC?,J/EXPCKR                       ;CK IF UNDERFL OR OVFL

                                      14D:    ;1****1101----------------------;RET FOR ADD/SUBF2, NORMALIZE AFTR ROUND
U 014D, 0600,003C,0180,F800,0080,C14C         D_D.RIGHT,SC_SC+1,J/ADDFDN         ;SHIFT RIGHT, ADD 1 TO EXP


                                                                                 ;RET FOR SRC=0 OR DEST=0, OR
                                      14E:    ;1****1110----------------------;RET FOR ADDF3/SUBF3
                                              EALU_SC,R(SP1)_PACK.FP,            ;PACK RESULT FOR *-R-R MODE
                                              SET.CC(INST),                      ;SET COND CODES
                                              CLR.IB0-1,PC_PC+2,                 ;UPDATE PC, POP IB
U 014E, 4008,D438,0180,F8C5,4070,05D1         SC?,J/EXPCKR                       ;CK IF UNDERFL OR OVFL

                                      14F:    ;1****1111----------------------;RET FOR ADD/SUBF3, NORMALIZE AFTR ROUND
U 014F, 0600,003C,0180,F800,0080,C14C         D_D.RIGHT,SC_SC+1,J/ADDFDN         ;SHIFT RIGHT, ADD 1 TO EXP
```

```
                                     ;ADDF3/SUBF3    ****, Register, Register
                                     ;ENTEP HERE AT B.FORK WITH D HAS SP1 OPERAND, LA HAS SP2 OPERAND.
                                     244:
                                                ;--------------------------------;
                                                SC_D(EXP)(B),                     ;GET EXP'S, SS
                                                FE_LA(EXP),SS_ALU15,              ;
U 0244, 001C,2038,0181,F800,099B,6044           CHK.FLT.OPR,CLK.UBCC,J/ADDF       ;LOAD ALU'S U BRANCH COND CODES



                                     ;ADDF3/SUBF3    ****, Short Literal, Register
                                     ;ENTER HERE AT B.FORK WITH D HAS SP1 OPERAND, Q SP2 OPERAND.
                                     240:
                                                ;--------------------------------;
                                                SC_D(EXP)(B),                     ;ADDF MEM MODE
                                                FE_Q(EXP),SS_ALU15,               ;
                                                D_Q,Q_D,                          ;SWAP D, Q
U 0240, 0C1D,2038,01E1,F800,019B,607C           CLK.UBCC                          ;LOAD ALU'S U BRANCH COND CODES

                                     =0****1100
                                                ;0****1100-----------------------;
                                                RC[T0]_D,D_D(FRAC),               ;D=SRC, RC[T0]=DST
                                                SC_NABS(SC-FE),                   ;
                                                SGN/ADD.SUB,                      ;SS_+/- INDICATOR, SD GETS DST SGN
                                                CHK.FLT.OPR,CLK.UBCC,             ;FAULT IF NEG FP 0
U 007C, 0901,123D,0186,F980,0890,E4F9           CALL,EALU?,J/ADDFX                ;

                                     =1****1100
                                                ;1****1100-----------------------;RETURN HERE WHEN RESULT = 0
                                                R(SP1)_K[Z[^O],                   ;
                                                EALU_K[ZERO],SET.CC(INST),        ;RESULT 0
U 017C, 4018,C038,1980,F8C5,4074,6062           CLR.IBO-1,PC_PC+2,J/IRD           ;GOTO NEXT INST

                                     =1****1110
                                     ADDFDB:    ;1****1110-----------------------;RETURN HERE WHEN DONE
                                                EALU_SC,R(SP1)_PACK.FP,           ;PACK RESULT
                                                SET.CC(INST),                     ;
                                                CLR.IBO-1,PC_PC+2,                ;
U 017E, 4008,D438,0180,F8C5,4070,05D1           SC?,J/EXPCKR                      ;CK IF UNDEPFL OR OVFL

                                                ;1****1111-----------------------;
U 017F, 0600,003C,0180,F800,0080,C17E           D_D.RIGHT,SC_SC+1,J/ADDFDB        ;SHIFT RIGHT, ADD 1 TO EXP
                                     =
                                     ;ADDF2/SUBF2    Register destination
                                     ;ENTER HERE AT B.FORK, WITH D CONTAINS SP1 OPERAND, AND LA SP2 OPERAND.
                                     22F:
                                                ;-------------------------------;
                                                SC_D(EXP)(B),FE_LA(EXP),        ;GET EXP'S
                                                SS_ALU15,                       ;
U 022F, 001C,2038,0181,F800,099B,603A           CHK.FLT.OPR,CLK.UBCC            ;

                                     =0****1010
                                                ;0****1010----------------------;2-OPR: UPC[1] IS DON'T CARE
                                                Q_D,RC[T0]_LA,                  ;SAVE DST OPD
                                                D_LA(FRAC),SC_NABS(SC-FE),      ;UNPACK DST FP, GET EXP DIFFERENCE
                                                SGN/ADD.SUB,                    ;SS_+/- INDICATOR, SET SD
                                                CHK.FLT.OPR,CLK.UBCC,           ;RSV OPD FAULT IF -0, SET ALUS CC
U 003A, 0900,123D,01E6,F980,0890,E4F9           CALL,EALU?,J/ADDFX              ;CHECK FOR 0 EXPS

                                     =1****1010
                                                ;-------------------------------;
                                                R(PRN)_K[ZERO],                 ;
                                                EALU_K[ZERO],SET.CC(INST),      ;RESULT 0
U 013A, C018,C038,1980,F8DC,4074,6062           CLR.IB.OPC,PC_PC+1,J/IRD        ;GOTO NEXT INST

                                     =1****1110
                                     ADDFDX:    ;1****1110----------------------;
                                                EALU_SC,R(PRN)_PACK.FP,         ;PACK RESULT
                                                SET.CC(INST),                   ;SET COND CODES
                                                CLR.IB.OPC,PC_PC+1,             ;UPDATE PC, POP IB
U 013E, C008,D438,0180,F8DC,4070,05E1           SC?,J/EXPCKP                    ;CK IF UNDERFL OR OVFL

                                                ;1****1111----------------------;
U 013F, 0600,003C,0180,F800,0080,C13E           D_D.RIGHT,SC_SC+1,J/ADDFDX      ;SHIFT RIGHT, ADD 1 TO EXP
                                     =
                                             ;10265  ;ADDF2/SUBF2     Memory destination
                                             ;10266  ;ADDF3/SUBF3     Register/Memory destination
                                             ;10267
                                             ;10268  ;ENTER HERE AT C.FORK
                                             ;10269  ;  TO MEMORY MODE INSTRUCTIONS
                                             ;10270  ;  WITH D CONTAINS DST (OR SP2), Q CONTAINS SRC (OR SP1) OPERANDS
                                             ;10271  38E:
                                             ;10272  ADDFA:   ;-------------------------------;
                                             ;10273           SC_Q(EXP)(B),FE_D(EXP),          ;ADDF MEM MODE
                                             ;10274           SS_ALU15,                        ;
U 038E, 001D,0038,0181,F800,099B,60D8        ;10275           CHK.FLT.OPR,CLK.UBCC             ;
                                             ;10276
                                             ;10277  =0****1000
                                             ;10278           ;0****1000----------------------;
                                             ;10279           RC[T0]_D,D_D(FRAC),              ;D=SRC, RC[T0]=DST, SRC-DST EXP
                                             ;10280           SC_NABS(SC-FE),                  ;
                                             ;10281           SGN/ADD.SUB,                     ;SS_+/- INDICATOR, SD GETS DST SGN
                                             ;10282           CHK.FLT.OPR,CLK.UBCC,            ;FAULT IF NEG FP 0
U 00D8, 0901,123D,0186,F980,0890,E4F9        ;10283           CALL,EALU?,J/ADDFX               ;
                                             ;10284
                                             ;10285  =1****1000
                                             ;10286  WR.Z:    ;1****1000----------------------;
                                             ;10287           EALU_K[ZERO],                    ;WRITE RESULT FLOAT 0
                                             ;10288           D_K[ZERO],SET.CC(INST),          ;
U 01D8, F818,C03B,19F0,F847,0074,6300        ;10289           WRITE.DEST,J/WRD                 ;
                                             ;10290
                                             ;10291  =1****1100
                                             ;10292  ADDFDA:  ;1****1100----------------------;RETURN HERE WHEN DONE
                                             ;10293           EALU_SC,D_PACK.FP,               ;ADDF2/SUBF2: PACK RESULT
                                             ;10294           SET.CC(INST),                    ;
U 01DC, 0808,D438,0180,F800,0070,05F1        ;10295           SC?,J/EXPCKM                     ;CK IF UNDERFL OR OVFL
                                             ;10296
                                             ;10297           ;1****1101----------------------;
U 01DD, 0600,003C,0180,F800,0080,C1DC        ;10298           D_D.RIGHT,SC_SC+1,J/ADDFDA       ;SHIFT RIGHT, ADD 1 TO EXP
                                             ;10299
                                             ;10300  ADDFDF:  ;1****1110----------------------;
                                             ;10301           EALU_SC,D_PACK.FP,               ;ADDF3/SUBF3: PACK RESULT
                                             ;10302           SET.CC(INST),                    ;
U 01DE, 0808,D438,0180,F800,0070,0601        ;10303           SC?,J/EXPCK                      ;CK IF UNDERFL OR OVFL
                                             ;10304
                                             ;10305           ;1****1111----------------------;
U 01DF, 0600,003C,0180,F800,0080,C1DE        ;10306           D_D.RIGHT,SC_SC+1,J/ADDFDE       ;SHIFT RIGHT, ADD 1 TO EXP
                                             ;10307  =
                                      ;10308  ;HERE BEGINS THE EXPONENT CHECKS OF FLOATING ADD/SUBTRACT
                                      ;10309  ;ENTER HERE BRANCHING ON EALU Z AND SC .NE. 0
                                      ;10310  ; WITH THE FIRST OPERAND EXPONENT IN SC AND PACKED IN Q,
                                      ;10311  ; THE SECOND OPERAND EXPONENT IN FE, AND UNPACKED FRACTION IN D.
                                      ;10312  ; SC SHOULD BE BEING LOADED WITH NABS(SC-FE).
                                      ;10313
                                      ;10314  =1001   ;-------------------------------;EALU Z=0, SC .EQL. 0 (SRC IS ZERO)
                                      ;10315  ADDFX:  ALU_Q,CHK.FLT.OPR,             ;CHECK FOR SRC RESERVED OPERAND
                                      ;10316          D_D.LEFT,SI/ZERO,             ;NORMALIZE FRACTION FROM DEST
                                      ;10317          SC_FE,                        ;GET DEST EXPONENT
U 04F9, 0501,203E,0180,F800,0881,010E ;10318          RETURN10E                     ;SEND BACK DEST AS RESULT
                                      ;10319
                                      ;10320          ;-------------------------------;EALU Z=0, SC .NEQ. 0 (NORMAL CASE)
                                      ;10321          R[R15]_Q,Q_Q(FRAC),           ;EXPS NE 0: UNPACK SRC FP
                                      ;10322          ID[T1]_D,                     ;SAVE DST FRAC
U 04FB, 0001,323C,C5C8,3EF8,0000,05A2 ;10323          EALU?,J/ADDFSH                ;COMPARE SRC - DST EXPS
                                      ;10324
                                      ;10325          ;-------------------------------;EALU Z=1, SC .EQL. 0 (BOTH ZERO)
                                      ;10326          ALU_Q,CHK.FLT.OPR,            ;MAKE SURE SRC ISN'T RESERVED
                                      ;10327          D_0,SC_K[ZERO],               ;CLEAR RESULT
U 04FD, 0F01,203E,1980,F800,0884,6100 ;10328          RETURN100
                                      ;10329
                                      ;10330          ;-------------------------------;EALU Z=1, SC .NEQ. 0 (DEST IS ZERO)
                                      ;10331          D_Q(FRAC),FE_Q(EXP),          ;UNPACK SRC AS RESULT
                                      ;10332          SD_SS,                        ; SRC SIGN XOR IR1 IS RESULT SIGN
U 04FF, 0901,203C,0184,F800,0108,64F9 ;10333          J/ADDFX                       ;NOW GO PACK IT UP AGAIN
```

```
                                         .TOC    "       F & D floating point   : ADDF/SUBF ROUTINE"

                                         ;ROUTINE SHARED BY DIFFERENT MODES OF ADDF, SUBF: ALSO POLYF AND ACBF
                                         ;
                                         ;ENTER HERE WITH :       D      = FRAC(DST)
                                         ;                        ID[T1] = FRAC(DST)
                                         ;                        FE     = EXP(DST)
                                         ;                        Q      = FRAC(SRC)
                                         ;                        R[15]  = SRC
                                         ;                        SC     = NABS (EXP DIFF)
                                         ;                        SS     = SIGN(SRC).xor.SIGN(DST).xor.IR<1>
                                         ;                        SD     = SIGN(DST)
                                         ;
                                         ; NOTE: EXP DIFFERENCE OF UP TO 31 IS SIGNIFICANT BECAUSE OF POLYF USAGE

                                         =0010
                                         ADDFSH: ;0010---------------------------; SRC.GTR.DST, SS = 0(ADD)
                                                 EALU_SC+K[.1F],CLK.UBCC,        ; SET UP EALU<N> TO CHECK SHIFT RANGE
U 05A2, 0000,003C,8DF8,F800,0014,85DD            Q_0,J/AL0                       ; Q GETS POSITIVE SIGN FOR DAL SHIFT

                                                 ;0011---------------------------; SRC.GTR.DST, SS = 1(SUB)
                                                 EALU_SC+K[.1F],CLK.UBCC,        ; SET UP EALU<N> TO CHECK SHIFT RANGE
U 05A3, 081F,2000,8D83,F800,0014,85E0            D_0-D,SD_NOT.SD,J/DF0           ; COMPLIMENT FRAC(DST) TO DO SUB

                                                 ;0110---------------------------; SRC.EQL.DST, SS = 0(ADD)
                                                 D_D+Q,Q_0,SC_FE,                ; ADD 2 ALIGNED FRACS
U 05A6, 081D,0014,01F8,F800,0091,05C3            CLK.UBCC,J/ADDFPK               ;

                                                 ;0111---------------------------; SRC.EQL.DST, SS = 1(SUB)
                                                 D_D-Q,Q_0,SC_FE,                ; SUBTRACT 2 ALIGNED FRACS
U 05A7, 081D,0000,01F8,F800,0091,0604            CLK.UBCC,J/NEGCK                ;

                                                 ;1010---------------------------; DST.GTR.SRC, SS = 0(ADD)
                                                 EALU_SC+K[.1F],CLK.UBCC,        ; SET UP EALU<N> TO CHECK SHIFT RANGE
U 05AA, 0C00,003C,8DF8,F800,0014,85E4            D_Q,Q_0,J/AL1                   ; D GETS SRC FRAC, CLR Q FOR DAL SHF

                                                 ;1011---------------------------; DST.GTR.SRC, SS = 1(SUB)
                                                 EALU_SC+K[.1F],CLK.UBCC,        ; SET UP EALU<N> TO CHECK SHIFT RANGE
U 05AB, 081F,0000,8D80,F800,0014,85ED            D_0-Q,J/DF1                     ; D GETS (-SRC FRAC) FOR SUBTRACT
                                         =
                                         AL0:    ;-------------------------------;
                                                 D_DAL.SC,                       ; ALIGN DST FRAC
                                                 Q_R[R15](FRAC),FE_R[R15](EXP),  ; Q, FE GET SRC FRAC,EXP
U 05DD, 0D00,123C,01C8,FA78,0108,6527            EALU?,J/DSTAP0                  ; GO CHECK SHIFT RANGE

                                         DF0:    ;-------------------------------;
U 05E0, 001B,0000,05C0,F800,0000,05DD            Q_0-K[.1],J/AL0                 ; SET Q TO ALL 1'S FOR SXT OF NEG NUMBER

                                         AL1:    ;-------------------------------;
                                                 D_DAL.SC,                       ; ALIGN SRC FRAC
                                                 Q_ID[T1],                       ; GET BACK DST FRAC
U 05E4, 0D00,123C,C5F0,2C00,0000,0537            EALU?,J/SRCAP0                  ; GO CHECK SHIFT RANGE

                                         DF1:    ;-------------------------------;
U 05ED, 001B,0000,05C0,F800,0000,05E4            Q_0-K[.1],J/AL1                 ; SET Q TO ALL 1'S FOR SXT OF NEG NUMBER
```

```
                                             =0111
                                             DSTAPO: ;0111-----------------------------; EALU<N> = 0, SHIFT WAS LEQ 31
                                                     D_D+Q,Q_0,SC_FE,                  ; ADD 2 ALIGNED FRACS
U 0527, 081D,0014,01F8,F800,0091,05C3                CLK.UBCC,J/ADDFPK                 ;

                                                     ;1111-----------------------------; EALU<N> = 1, SHIFT WAS GTR 31
                                                     ALU_Q+K[ZERO],                    ;
                                                     D_ALU.LEFT,SI/ZERO,               ; DST APPROX 0 W.R.T. SRC
U 052F, 0839,2014,1980,F800,0091,0621                CLK.UBCC,SC_FE,J/ROUND1           ; CLR ALU C31

                                             =0111
                                             SRCAPO: ;0111-----------------------------; EALU<N> = 0, SHIFT WAS LEQ 31
                                                     D_D+Q,Q_0,SC_FE,                  ;
U 0537, 081D,0014,01F8,F800,0091,05C3                CLK.UBCC,J/ADDFPK                 ; ADD 2 ALIGNED FRACS

                                                     ;1111-----------------------------; EALU<N> = 1, SHIFT WAS GTR 31
U 053F, 0821,203C,4180,F800,0081,05F2                SC_FE,D_Q.LEFT,SI/ZERO,K[.80]     ; SRC APPROX 0 WRT DST, SET CC

                                                     ;---------------------------------;
U 05F2, 0819,0014,4180,F800,0010,0621                D_D+K[.80],CLK.UBCC,J/ROUND1      ; THE ROUNDING IS FOR POLYF


                                             NEGCK:  ;---------------------------------;
U 0604, 001B,1B10,0580,F800,0010,05C3                ALU_0+K[.1]+1,CLK.UBCC,ALU?       ; CK IF NEG RESULT
                                             =0011
                                             ADDFPK: ;0011-----------------------------;
                                                     SC_SC-SHF.VAL,D_DAL.NORM,         ; NORMALIZE FRAC
U 05C3, 0E00,003C,4180,F800,008C,A611                K[.80],J/ROUND                    ; SET UP FOR CONST -81 VIA MASK

                                                     ;0111-----------------------------;
                                                     D_K[ZERO],SC_K[ZERO],             ; RESULT 0
U 05C7, 0818,003A,1980,F800,0084,6100                RETURN100                         ;

                                                     ;1011-----------------------------;
U 05CB, 081F,2000,0183,F800,0000,05C3                D_0-D,SD_NOT.SD,J/ADDFPK          ; GET ABS(DIFF) FOR FRAC
                                             =

                                             ROUND:  ;---------------------------------;
U 0611, 0819,0014,4180,F800,0090,C621                D_D+K[.80],SC_SC+1,CLK.UBCC       ; D_D+80 FOR ROUNDING

                                             ROUND1: ;---------------------------------;
                                                     ALU_R(SP1)+K[ZERO].RLOG,          ; SAVE REG # FOR R-R OVERFLOWS
U 0621, 0018,1B1A,1980,F840,0000,010C                ALU?,RETURN10C                    ; CHECK IF RENORMALIZE AGAIN
```

```
                                       ;10432  ;CLEANUP FOR SL/R-R, *-SL/R-R ADDS. RESULT ALREADY STORED.
                                       ;10433  ;COME HERE WITH BEN/SC TO TEST FOR OVERFLOW AND UNDERFLOW.
                                       ;10434  ;THE REGISTER NUMBER OF THE DESTINATION IS LOST HERE, AND MUST
                                       ;10435  ;BE RECONSTRUCTED FROM THE RLOG STACK IF OVER/UNDERFLOW EXISTS.
                                       ;10436  =0001
                                       ;10437  EXPCKR: ;0001---------------------------;
                                       ;10438          RC[TO]_K[ZERO],                   ;ZERO EXP, UNDERFLOW, SET TRAP STACK
U 05D1, 0018,0039,1980,F980,0000,0E09  ;10439          CALL,J7UNDRFL                     ;SET CES FOR UNDERFLOW
                                       ;10440
                                       ;10441          ;0011---------------------------;
U 05D3, F80C,003B,01F1,F857,139B,6000  ;10442          IRD                               ;01 TO FF FOR EXP, OK
                                       ;10443
                                       ;10444          ;0101---------------------------;
                                       ;10445          RC[TO]_K[ZERO],                   ;NEG EXP, UNDERFLOW, SET TRAP STACK
U 05D5, 0018,0039,1980,F980,0000,0E09  ;10446          CALL,J7UNDRFL                     ;SET CES FOR UNDERFLOW
                                       ;10447
                                       ;10448          ;0111---------------------------;
                                       ;10449          RC[TO]_K[.8000],                  ;> FF EXP, OVERFLOW, SET TRAP STACK
U 05D7, 0018,0039,4580,F980,0000,0E03  ;10450          CALL,J7OVFL                       ;SET CES FOR OVERFLOW
                                       ;10451  =1111
                                       ;10452          ;1111---------------------------;
U 05DF, 0840,0038,0180,F800,1C00,0625  ;10453          D_RLOG.RIGHT                      ;RETURN HERE FOR UNDERFLOW, OVERFLOW
                                       ;10454
                                       ;10455          ;-------------------------------;
U 0625, 0811,0038,0180,F900,0088,6636  ;10456          SC_D(EXP)(A), D_RC[TO]            ; PUT REG # IN SC, GET RESULT
                                       ;10457
                                       ;10458          ;-------------------------------;
U 0636, 0001,003C,0180,F8E8,0000,0062  ;10459          R(SC)_D, J/IRD                    ;  STORE IT AND GET OUT
                                       ;10460
                                       ;10461
                                       ;10462  ;CLEANUP FOR *-R ADDS. RESULT ALREADY STORED IN R(PRN).
                                       ;10463  ;COME HERE WITH BEN/SC TO TEST FOR UNDERFLOW AND OVERFLOW.
                                       ;10464  =0001
                                       ;10465  EXPCKP: ;0001---------------------------;
                                       ;10466          R(PRN)_K[ZERO],                   ;ZERO EXP, UNDERFLOW, SET TRAP STACK
U 05E1, 0018,0039,1980,F8D8,0000,0E09  ;10467          CALL,J7UNDRFL                     ;SET CES FOR UNDERFLOW
                                       ;10468
                                       ;10469          ;0011---------------------------;
U 05E3, F80C,003B,01F1,F857,139B,6000  ;10470          IRD                               ;01 TO FF FOR EXP, OK
                                       ;10471
                                       ;10472          ;0101---------------------------;
                                       ;10473          R(PRN)_K[ZERO],                   ;NEG EXP, UNDERFLOW, SET TRAP STACK
U 05E5, 0018,0039,1980,F8D8,0000,0E09  ;10474          CALL,J7UNDRFL                     ;SET CES FOR UNDERFLOW
                                       ;10475
                                       ;10476  OVFLP:  ;0111---------------------------;
                                       ;10477          R(PRN)_K[.8000],                  ;> FF EXP, OVERFLOW, SET TRAP STACK
U 05E7, 0018,0039,4580,F8D8,0000,0E03  ;10478          CALL,J7OVFL                       ;SET CES FOR OVERFLOW
                                       ;10479  =1111
                                       ;10480          ;1111---------------------------;
U 05EF, F80C,003B,01F1,F857,139B,6000  ;10481          IRD                               ;RETURN HERE FOR UNDERFLOW, OVERFLOW
```

```
                                          ;CLEANUP FOR *-* ADDS, RESULT IN D, ADDRESS IN VA.
                                          ;COME HERE WITH BEN/SC TO TEST FOR UNDERFLOW AND OVERFLOW.
                                          =0001
                                          EXPCKM: ;0001--------------------------------;
                                                  D_0,                                  ;ZERO EXP, UNDERFLOW, SET TRAP STACK
U 05F1, 0F00,003D,0180,F800,0000,0E09             CALL,J/UNDRFL                         ;GOTO SET CES FOR UNDERFLOW

                                                  ;0011--------------------------------;
                                                  CACHE_D[INST.DEP],                    ;
U 05F3, C000,C03C,0180,3004,4000,0062             CLR.IB.OPC,PC_PC+1,J/IRD              ;01 TO FF FOR EXP, OK

                                                  ;0101--------------------------------;
                                                  D_0,                                  ;NEG EXP, UNDERFLOW, SET TRAP STACK
U 05F5, 0F00,003D,0180,F800,0000,0E09             CALL,J/UNDRFL                         ;GOTO SET CES FOR UNDERFLOW

                                                  ;0111--------------------------------;
                                                  D_K[.8000],                           ;> FF EXP, OVERFLOW, SET TRAP STACK
U 05F7, 0818,0039,4580,F800,0000,0E03             CALL,J/OVFL                           ;GOTO SET CES FOR OVERFLOW
                                          =1111
                                          DEST:   ;1111--------------------------------;
                                                  CACHE_D[INST.DEP],                    ;WRITE DEST FOR UNDERFLOW, OVERFLOW
U 05FF, C000,C03C,0180,3004,4000,0062             CLR.IB.OPC,PC_PC+1,J/IRD



                                          ;CLEANUP FOR *-*-* ADDS, RESULT IN D, DEST NOT EVALUATED YET.
                                          ;COME HERE WITH BEN/SC TO TEST FOR UNDERFLOW AND OVERFLOW.
                                          =0001
                                          EXPCK:  ;0001--------------------------------;
                                                  D_0,                                  ;ZERO EXP, UNDERFLOW, SET TRAP STACK
U 0601, 0F00,003D,0180,F800,0000,0E09             CALL,J/UNDRFL                         ;GOTO SET CES FOR UNDERFLOW

                                          WRDST:  ;0011--------------------------------;
U 0603, F000,003F,01F0,F847,0000,0300             WRITE.DEST                            ;01 TO FF FOR EXP, OK

                                                  ;0101--------------------------------;
                                                  D_0,                                  ;NEG EXP, UNDERFLOW, SET TRAP STACK
U 0605, 0F00,003D,0180,F800,0000,0E09             CALL,J/UNDRFL                         ;GOTO SET CES FOR UNDERFLOW

                                          OVFLK:  ;0111--------------------------------;
                                                  D_K[.8000],                           ;> FF EXP, OVERFLOW, SET TRAP STACK
U 0607, 0818,0039,4580,F800,0000,0E03             CALL,J/OVFL                           ;GOTO SET CES FOR OVERFLOW
                                          =1111
                                                  ;1111--------------------------------;
U 060F, F000,003F,01F0,F847,0000,0300             WRITE.DEST                            ;WRITE DEST FOR UNDERFLOW, OVERFLOW
```

```
                                          .TOC     "       F & D floating point   : MULF"

                                          ;MULF, *-R
                                          ;ENTER HERE AT B.FORK WITH LA CONTAINS DST, D CONTAINS SRC.

                                          220:      ;-------------------------------;
                                          MULF2:    Q_D(FRAC)(B),                   ;GET SRC FRAC
                                                    SC_D(EXP)(B),                   ;GET SRC EXP
                                                    FE_LA(EXP),                     ;GET DST EXP
                                                    SS_ALU15,CHK.FLT.OPR,           ;SS GETS SRC SIGN, CHK -0,
U 0220, 001C,2038,01C9,F800,099B,664C               CLK.UBCC                        ;CLOCK DST EXP

                                                    ;-------------------------------;
                                                    RC[T0]_Q,FE_SC+FE,              ;PROD EXP = SUM OF EXP'S
U 064C, 0001,323C,0180,F980,0100,8069               EALU?                           ;DST EXP (EALU.Z), SRC EXP (SC) = 0?
                                          =01001
                                          MULF.0:   ;01001--------------------------;
                                                    R(PRN)_K[ZERO],SET.CC(INST),    ;PROD = 0: SET COND CODES
U 0069, C018,C038,1980,F8DC,4070,0062               CLR.IB.OPC,PC_PC+1,J/IRD        ;

                                                    ;01011--------------------------;
                                                    D_LA(FRAC),                     ;GET DST FRAC
                                                    LC_RC[T0],Q_0,                  ;
                                                    SC_K[.FFF9],                    ;GET SHIFT VALUE -7
                                                    SS_SS.XOR.ALU15&SD_ALU15,       ;GET RESULTANT SIGN TO SS
U 006B, 0900,003D,BDFD,F900,0084,665D               CALL,J/MULFX                    ;

                                          ;*********************************************
                                          ;* Patch no. 088, PCS 006B trapped to WCS 1198 *
                                          ;*********************************************

                                                    ;01101--------------------------;
                                                    ALU_LA,CHK.FLT.OPR,             ;
U 006D, 0000,003C,0180,F800,0800,0069               J/MULF.0                        ;

                                                    ;01111--------------------------;
                                                    ALU_LA,CHK.FLT.OPR,             ;
U 006F, 0000,003C,0180,F800,0800,0069               J/MULF.0                        ;

                                          =11011    ;11011--------------------------;
U 007B, 0000,1B3C,0180,F800,0000,013E               ALU?, J/ADDFDX                  ;RENORMALIZE (IF NEEDED), PACK & STORE
                                          =
                                           ;10569  ;MULF, *-* OR *-*-*
                                           ;10570  ;ENTER HERE WITH D CONTAINS DST, Q CONTAINS SRC
                                           ;10571  038F:
                                           ;10572  MULF:     ;-----------------------------------;
                                           ;10573            Q_Q(FRAC)(B),SC_Q(EXP)(B),          ;GET SRC FRAC, EXP
                                           ;10574            FE_D(EXP),                          ;
U 038F, 001D,0038,01C9,F800,099B,6651      ;10575            SS_ALU15,CHK.FLT.OPR,CLK.UBCC       ;
                                           ;10576
                                           ;10577            ;-----------------------------------;
                                           ;10578            RC[T0]_Q,                           ;GET 2 TIMES MULTI'CAND TO LC
                                           ;10579            FE_SC+FE,                           ;
U 0651, 0001,323C,0180,F980,0100,81E9      ;10580            EALU?                               ;
                                           ;10581
                                           ;10582  =01001    ;01001------------------------------;
                                           ;10583            EALU_K[ZERO],D_K[ZERO],             ;
                                           ;10584            SET.CC(INST),                       ;WRITE RESULT FLOAT 0
U 01E9, F818,C03B,19F0,F847,0074,6300      ;10585            WRITE.DEST,J/WRD                    ;
                                           ;10586
                                           ;10587            ;01011------------------------------;
                                           ;10588            D_D(FRAC),LC_RC[T0],Q_0,            ;GET DST FRAC
                                           ;10589            SC_K[.FFF9],                        ;GET SHIFT VALUE -7
                                           ;10590            SS_SS.XOR.ALU15&SD_ALU15,           ;GET RESULTANT SIGN TO SS
U 01EB, 0901,003D,BDFD,F900,0084,665D      ;10591            CALL,J/MULFX                        ;
                                           ;10592
                                           ;10593            ;01101------------------------------;
                                           ;10594            ALU_D,CHK.FLT.OPR,D_0,              ;
U 01ED, 0F01,003C,0180,F800,0800,01D8      ;10595            J/WR.Z                              ;
                                           ;10596
                                           ;10597            ;01111------------------------------;
                                           ;10598            ALU_D,CHK.FLT.OPR,D_0,              ;
U 01EF, 0F01,003C,0180,F800,0800,01D8      ;10599            J/WR.Z                              ;
                                           ;10600
                                           ;10601  =11011    ;11011------------------------------;
U 01FB, 0000,1B3C,0180,F800,0000,01DC      ;10602            ALU?, J/ADDFDA                      ;RENORMALIZE (IF NEEDED), PACK & STORE
                                           ;10603  =
                                    ;FRACTION MULTIPLY ROUTINE - USES INTEGER MULTIPLY SUBR
                                    ;
                                    ;INPUTS:         SC = -7, D = MULTIPLIER FRACTION, SS = RESULT SIGN,
                                    ;                FE HAS SUM OF INPUT EXPS, LC HAS 2 * MULTIPLICAND FRACTION,
                                    ;
                                    ;OUTPUTS:        D = ROUNDED PRODUCT FRACTION,
                                    ;                SC = PRODUCT EXPONENT,
                                    ;                SS & SD = PRODUCT SIGN
                                    ;
                                    ;TEMPORARIES:    R[R15], LA, LB, Q
                                    ;
                                    MULFX:  ;---------------------------------------;
                                            D_DAL.SC,                               ;SHIFT M'IER READY, R[R15]_M'CAND
                                            R[R15]_LC.RIGHT,SI/ZERO,                ;
U 065D, 0D50,0038,8584,FAF8,0084,6398       SC_K[.C],SD_SS                          ;SET LOOP CT FOR 26. BITS

                                    =0*     ;0*-------------------------------------;
                                            ALU_O(A),                               ;LB_M'CAND
                                            D_D.RIGHT2,SI/ZERO,LAB_R[R15],          ;
U 0398, 0203,0C3D,0180,FA78,0000,0350       CALL,BEN/MUL,J/MULPP                    ;CALL MULTIPLICATION ROUTINE

                                            ;1*-------------------------------------;
U 039A, 0C18,0038,41E0,FAF8,0081,0664       SC_FE,D_Q,Q_D,R[R15]_K[.80]             ;ALWAYS POS PROD

                                            ;---------------------------------------;
                                            SC_SC-SHF.VAL,D_DAL.NORM,               ;SHIFT LEFT JUSTIFIED
U 0664, 0E00,003C,0180,FA78,008C,A669       LAB_R[R15]                              ;

                                            ;---------------------------------------;
                                            D_D+LB,CLK.UBCC,SC_SC-K[.7C],           ;ROUNDING
U 0669, 080D,0016,9D80,F800,0094,A010       RETURN10                                ;
```

```
                                        ;10635  .TOC    "       F & D floating point   : DIVF"
                                        ;10636
                                        ;10637  ;DIVF: *-R
                                        ;10638  ;ENTER HERE AT B.FORK WITH LA CONTAINS DST (D'END), D CONTAINS SRC (D'SOR)
                                        ;10639  221:
                                        ;10640  DIVF2:  ;---------------------------------;
                                        ;10641          Q_D(FRAC)(B),SC_D(EXP)(B),        ;GET SRC FRACTION
                                        ;10642          CLK.UBCC,
                                        ;10643          FE_LA(EXP),                       ;CK IF NEG FPO
U 0221, 001C,2038,01C9,F800,099B,667E   ;10644          SS_ALU15,CHK.FLT.OPR              ;
                                        ;10645
                                        ;10646          ;---------------------------------;
U 067E, 0001,323C,0180,F980,0181,0589   ;10647          SC_FE,FE_SC,RC[T0]_Q,EALU?        ;SWAP EXPS
                                        ;10648
                                        ;10649  ; ***********************************************
                                        ;10650  ; * Patch no. 087, PCS 067E trapped to WCS 1197 *
                                        ;10651  ; ***********************************************
                                        ;10652
                                        ;10653  =1001   ;1001---------------------------;
U 0589, 0000,003C,0180,F800,0800,03D0   ;10654          ALU_LA,CHK.FLT.OPR,J/DIVF6        ;D'SOR = 0:  NO DIVIDE
                                        ;10655
                                        ;10656          ;1011---------------------------;
U 058B, 0000,003C,0180,F800,0100,A3BC   ;10657          FE_SC-FE,J/DIVF3                  ;D'SOR TO LB
                                        ;10658
                                        ;10659  ; ***********************************************
                                        ;10660  ; * Patch no. 022, PCS 058B trapped to WCS 1159 *
                                        ;10661  ; ***********************************************
                                        ;10662
                                        ;10663          ;1101---------------------------;
U 058D, 0000,003C,0180,F800,0800,03D0   ;10664          ALU_LA,CHK.FLT.OPR,J/DIVF6        ;D'SOR = 0:  NO DIVIDE
                                        ;10665
                                        ;10666          ;1111---------------------------;
U 058F, 0000,003C,0180,F800,0800,0069   ;10667          ALU_LA,CHK.FLT.OPR,J/MULF.0       ;D'END = 0
                                        ;10668  =0
                                        ;10669  DIVF3:  ;0------------------------------;
                                        ;10670          D_LA(FRAC),                       ;GET D'END FRAC, RESULT SIGN
                                        ;10671          SS_SS.XOR.ALU15&SD_ALU15,         ;
                                        ;10672          LC_RC[T0],CHK.FLT.OPR,            ;D'SOR TO LB, SET LOOP CT FOR 25.
U 03BC, 0900,003D,B985,F900,0884,6327   ;10673          SC_K[.19],CALL,J/DIVFX            ;
                                        ;10674
                                        ;10675          ;1------------------------------;
U 03BD, 0000,1B3C,0180,F800,0000,013E   ;10676          ALU?, J/ADDFDX                    ;TEST FOR NORM, PACK & STORE
                                        ;10677  =0
                                        ;10678  DIVF6:  ;0------------------------------;
U 03D0, 0818,0039,4580,F800,0000,0E00   ;10679          D_K[.8000],CALL,J/DIVBYO          ;SET CES FOR FL DIV BY 0
                                        ;10680
                                        ;10681          ;1------------------------------;
                                        ;10682          R(PRN)_K[.8000],                  ;
U 03D1, C018,0038,4580,F8DC,4000,0062   ;10683          CLR.IB.OPC,PC_PC+1,J/IRD          ;
```

```
                                          ;DIVF: *-*, *-*-*
                                          ;ENTER HERE WITH D CONTAINS DST, Q CONTAINS SRC
                                          038D:
                                          DIVF:     ;-------------------------------;
                                                    Q_Q(FRAC)(B),                   ;GET SRC FRACTION
                                                    SC_Q(EXP)(B),CLK.UBCC,          ;
U 038D, 001D,0038,01C9,F800,099B,6683               FE_D(EXP),SS_ALU15,CHK.FLT.OPR  ;CK IF NEG FPO

                                                    ;-------------------------------;
U 0683, 0001,323C,0180,F980,0181,0619               SC_FE,FE_SC,RC[T0]_Q,EALU?      ;SWAP EXPS

                                          =1001     ;1001---------------------------;
U 0619, 0001,003C,0180,F800,0800,03E0               ALU_D,CHK.FLT.OPR,J/DIVF8       ;D'SOR = 0:  NO DIVIDE

                                                    ;1011---------------------------;
U 061B, 0000,003C,0180,F800,0100,A3D8               FE_SC-FE,J/DIVF4                ;D'SOR TO LB

                                          ; ***********************************************
                                          ; * Patch no. 021, PCS 061B trapped to WCS 1158 *
                                          ; ***********************************************

                                                    ;1101---------------------------;
U 061D, 0001,003C,0180,F800,0800,03E0               ALU_D,CHK.FLT.OPR,J/DIVF8       ;D'SOR = 0:  NO DIVIDE

                                                    ;1111---------------------------;
                                                    ALU_D,CHK.FLT.OPR,D_0,          ;D'END = 0, THEREFORE RESULT 0
U 061F, 0F01,003C,0180,F800,0800,01D8               J/WR.Z
                                          =0
                                          DIVF4:    ;0------------------------------;
                                                    D_D(FRAC),                      ;GET D'END FRAC, RESULT SIGN
                                                    SS_SS.XOR.ALU15&SD_ALU15,       ;
                                                    LC_RC[T0],CHK.FLT.OPR,          ;D'SOR TO LB, SET LOOP CT FOR 25.
U 03D8, 0901,003D,B985,F900,0884,6327               SC_K[.19],CALL,J/DIVFX          ;

                                                    ;1------------------------------;
U 03D9, 0000,1B3C,0180,F800,0000,01DC               ALU?, J/ADDFDA                  ;TEST FOR NORM, PACK & STORE
```

```
                                   ;        FRACTION DIVIDE ROUTINE - RESTORING METHOD.  ENTER AT DIVFX
                                   ;
                                   ;INPUTS:          D = DIVIDEND FRACTION
                                   ;                 LC = RC[T0] = DIVISOR FRACTION
                                   ;                 SS = QUOTIENT SIGN
                                   ;                 SC = 19 (HEX)
                                   ;                 FE = DIFFERENCE IN INPUT EXPONENTS
                                   ;                 Q<6:0> = 0
                                   ;
                                   ;OUTPUTS:         D = ROUNDED QUOTIENT FRACTION
                                   ;                 SS = SD = QUOTIENT SIGN
                                   ;                 SC = BIASED QUOTIENT EXPONENT
                                   ;
                                   ;TEMPORARIES:     R[R15], LA, LB, Q

                                   =011
                                   DIVF0:  ;011-----------------------------;
                                           R[R15]_K[.80],                   ;SET UP FOR PACK FP
                                           D_Q,Q_0,SC_SC+FE,SD_SS,          ;
U 0323, 0C18,0038,41FC,FAF8,0080,8694      J7DIVF1                          ;SC HAS -1 (SHOULD HAVE
                                                                            ;K[.87] IN NEXT STATE)

                                   DIVFX:  ;111-----------------------------;
                                           ALU_D-LC,                        ;
                                           DK/DIV,Q_Q.LEFT,                 ;
                                           SHF/LEFT,SI/DIV,                 ;
U 0327, 0431,0C00,06A8,F800,0084,A323      SC_SC-K[.1],MUL? J/DIVF0         ;LOOP FOR DIV

                                   DIVF1:  ;--------------------------------;
                                           SC_SC-SHF.VAL,D_DAL.NORM,        ;NORMAILIZE FIRST
U 0694, 0E00,003C,0180,FA78,008C,A6A6      LAB_R[R15]                       ;

                                           ;--------------------------------;
                                           D_D+LB,CLK.UBCC,SC_SC+K[.88],    ;ROUNDING, AND ADD 80 TO ADJUST EXP
U 06A6, 080D,0016,C580,F800,0094,8001      RETURN1
                                   =0
                                   DIV 8:  ;0-------------------------------;
U 03E0, 0818,0039,4580,F800,0000,0E00      D_K[.8000],CALL,J/DIVBY0         ;SET CES FOR FL DIV BY 0

                                           ;1-------------------------------;
U 03E1, F000,003F,01F0,F847,0000,0300      WRITE.DEST,J/WRD                 ;
```

```
                                        .TOC    "       F & D floating point   : CMPD"

                                        ;DOUBLE PRECISION FLOATING POINT CMP.
                                        ;ENTER HERE WITH SRC<H,L> = RC[T0], Q;  DST<H,L> = RC[T1], D.

                                        ;ALGORITHM:
                                        ;       SIMILIAR TO CMPF, WITH THE FOLLOWING SET OF TESTS BEING USED
                                        ;       FOR MAGNITUDE COMPARISON:
                                        ;       1.      EXPONENTS
                                        ;       2.      FRACTION<0:23> (BIT 0 IS MOST SIGNIFICANT IN THIS NOTATION)
                                        ;       3.      FRACTION<24:39>
                                        ;       4.      FRACTION<40:55>

                                        ;       *** NOTE ***
                                        ;       THIS ROUTINE HAS A MAJOR BUG, STARTING AT THE MICROINSTRUCTION
                                        ;       AFTER THE ONE LABELLED "GETL1" AND CONTINUING TO THE END OF THE
                                        ;       INSTRUCTION.  SEE THE ECO LISTING FOR CORRECTIONS.

                                        382:    ;-------------------------------;
                                        CMPD:   ID[T1]_D, D_Q, ALU_RC[T1],      ;SAVE DST<L>, GET DST<H>
                                                SC_ALU(EXP), Q_ALU(FRAC),       ;UNPACK DST<H>
U 0382, 0C10,0038,C5C9,3D08,0883,06B1           SS_ALU15, CHK.FLT.OPR           ;AND CHECK FOR -0
                                                ;-------------------------------;
                                                ID[T2]_D, ALU_RC[T0], FE_SC,    ;SAVE SRC<L>, GET SRC<H>
                                                D_ALU(FRAC),
                                                SC_ALU(EXP), CLK.UBCC,          ;UNPACK SRC<H>, SET CC ON EXPS
                                                SS_SS.XOR.ALU15&SD_ALU15,       ;SS=0 IF SIGNS EQUAL
U 06B1, 0910,1438,C985,3D00,0993,02E1           CHK.FLT.OPR, SC.GT.0?           ;TEST FOR -0 SRC & 0 DST
                                                ;-------------------------------;
                                        =*01
                                                ALU_RC[T0],SET.CC(INST),
                                                CLR.IB.OPC,PC_PC+1,             ;DST = 0, CC SET BY SRC
U 02E1, C010,C038,0180,F904,4070,0062           J/IRD
                                                ;-------------------------------;
                                                ALU_D-Q,EALU_SC-FE,CLK.UBCC,    ;SRC - DST FOR FRAC, EXPS
                                                Q_ID[T1], LC_RC[T1],            ;GET DST<L> IN Q, SRC<H> IN LC
U 02E3, 001D,1200,C5F0,2D08,0010,A432           EALU?                           ;BEN ON SS - EALU N&Z KNOWN 0
                                                ;-------------------------------;
                                        =**10
                                                R[R15]_LC, D_Q, Q_ID[T2],       ;R[R15] DST<H>, D=DST<L>, Q=SRC<L>
U 0432, 0C10,1238,C9F0,2EF8,0000,0623           EALU?,J/GETL1                   ;COMPARE SRC - DST EXPS
                                                ;-------------------------------;
U 0433, 0010,0038,0180,FAF8,0000,065A           R[R15]_LC, J/CHECKF             ;SGNS DIFF - SET CC'S FROM -DST
                                                                                ;(CANT SET FROM SRC, MAY BE 0)
                                                ;-------------------------------;
                                        =0011
                                        GETL1:  ALU_RC[T0],SET.CC(INST),
U 0623, C010,C038,0180,F904,4070,0062           CLR.IB.OPC,PC_PC+1,J/IRD        ;SRC(EXP) > DST(EXP), CC_SRC
                                                ;-------------------------------;
U 0627, 001D,7B00,0180,F800,0000,0633           ALU_Q-D, WORD, ALU?, J/CHECKH   ;CHECK SRC - DST FRAC<H>'S **SEE ECO**

                                        ; ************************************************
                                        ; * Patch no. 008, PCS 0627 trapped to WCS 1148 *
                                        ; ************************************************
```

```
                                                          ;--------------------------------;
                                                          ALU_R[R15].XOR.K[.8000],
                                                          SET.CC(INST),                      ;DST(EXP) > SRC(EXP), CC_-DST
U 062B, C018,C020,4580,FA7C,4070,0062                     CLR.IB.OPC,PC_PC+1,J/IRD
                                                =;END     ;--------------------------------;

                                                =0011
                                                CHECKH:   ALU_RC[T0],SET.CC(INST),
                                                          CLR.IB.OPC,PC_PC+1,                ;SRC<H> > DST<H>, CC_SRC
U 0633, C010,C038,0180,F904,4070,0062                     J/IRD
                                                          ;--------------------------------;

U 0637, 001D,3B00,0180,F800,0000,0643                     ALU_Q-D, LONG, ALU?, J/CHECKL      ;SRC<H>=DST<H> - TEST SRC<M>-DST<M>
                                                          ;--------------------------------;
                                                          ALU_R[R15].XOR.K[.8000],
                                                          SET.CC(INST),                      ;DST<H> > SRC<H>, CC_-DST
U 063B, C018,C020,4580,FA7C,4070,0062                     CLR.IB.OPC,PC_PC+1,J/IRD
                                                =;END     ;--------------------------------;

                                                =0011
                                                CHECKL:   ALU_RC[T0],SET.CC(INST),
                                                          CLR.IB.OPC,PC_PC+1,                ;SRC<M> > DST<M>, CC_SRC
U 0643, C010,C038,0180,F904,4070,0062                     J/IRD
                                                          ;--------------------------------;
U 0647, 0000,1B3C,0180,F800,0000,065A                     ALU?, J/CHECKF                     ;SRC<M>=DST<M> - TEST SRC<L>-DST<L>
                                                          ;--------------------------------;
                                                          ALU_R[R15].XOR.K[.8000],
                                                          SET.CC(INST),                      ;DST<M> > SRC<M>, CC_-DST
U 064B, C018,C020,4580,FA7C,4070,0062                     CLR.IB.OPC,PC_PC+1,J/IRD
                                                =;END     ;--------------------------------;
                                                =1010
                                                CHECKF:   ALU_R[R15].XOR.K[.8000],
                                                          SET.CC(INST),                      ;DST<L> > SRC<L>, CC_-DST
U 065A, C018,C020,4580,FA7C,4070,0062                     CLR.IB.OPC,PC_PC+1,J/IRD           ;UPDATE PC, POP IB FOR NXT INST
                                                          ;--------------------------------;
                                                          ALU_RC[T0],SET.CC(INST),           ;SRC<L> > DST<L>, SRC SET CC
U 065B, C010,C038,0180,F904,4070,0062                     CLR.IB.OPC,PC_PC+1,J/IRD
                                                          ;--------------------------------;
                                                =1111     ALU_K[ZERO],SET.CC(INST),          ;SRC<L> = DST<L>
U 065F, C018,C038,1980,F804,4070,0062                     CLR.IB.OPC,PC_PC+1,J/IRD           ;UPDATE PC, POP IB, GOTO NXT INST
                                                          ;--------------------------------;
                                                .TOC    ''       F & D floating point  : UNPACK DOUBLE OPERANDS''

                                                ;ROUTINE TO UNPACK DOUBLE FLOATING POINT OPERANDS.
                                                ;INPUTS:         RC[T0] = OPERAND 1 HIGH LONGWORD
                                                ;                Q = RC[T3] = OPERAND 1 LOW LONGWORD
                                                ;                D = OPERAND 1 LOW LONGWORD, WITH HIGH AND LOW WORD SWAPPED
                                                ;                RC[T1] = OPERAND 2 HIGH LONGWORD
                                                ;                RC[T6] = OPERAND 2 LOW LONGWORD
                                                ;                SC = -7
                                                ;                IR<2:1> = TYPE OF OPERATION BEING PERFORMED:
                                                ;                00 = ADD, 01 = SUB, 10 = MUL, 11 = DIV

                                                ;                (STARTING WITH THE OPERANDS IN <RC[T0],Q> AND <RC[T1],D>,
                                                ;                THE FOLLOWING SEQUENCE OF STATES SETS THINGS UP RIGHT:
                                                ;        1:      RC6_D, D_Q, SC_16.
                                                ;=00     2:      D_DAL.SC, RC3_D, SC_-7, CALL UNPACK )

                                                ;OUTPUTS:
                                                ;                ID[T0] = OPERAND 1 FRACTION <H>
                                                ;                Q = ID[T2] = OPERAND 1 FRACTION <L>
                                                ;                RC[T5] = ID[T1] = OPERAND 2 FRACTION <H>
                                                ;                D = OPERAND 2 FRACTION <L>
                                                ;                SC = OPERAND 2 EXPONENT
                                                ;                FE = NABS(EXPONENT DIFFERENCE) IF IR<2:1> = 00 OR 01,
                                                ;                   = SUM OF EXPONENTS IF IR<2:1> = 10,
                                                ;                   = (EXPONENT OF OPERAND 1) - (EXPONENT OF OPERAND 2) IF IR<2:1>=11
                                                ;                SS = (SIGN OF OPERAND 1) .XOR. (SIGN OF OPERAND 2) .XOR. IR<1>
                                                ;                RC[T0], RC[T1], RC[T3], RC[T6] PRESERVED

                                                ;TEMPORARIES:    ID[T3] HOLDS OPERAND 1 <L>,
                                                ;                R[R15] HOLDS OPERAND 1 <H>,
                                                ;                LA, LB, LC HOLD VARIOUS THINGS

                                                ;RETURNS:        RETURN @ 1 IF OPERAND 1 = 0
                                                ;                RETURN @ 2 IF OPERAND 1 <> 0 , OPERAND 2 = 0
                                                ;                RETURN @ 3 IF BOTH OPERANDS ARE NON-ZERO

                                                UNPACK: ;-----------------------;
                                                        LC_RC[T0],              ;       LATCH SRC0 <H>
U 06CA, 0000,003C,5D80,F900,0104,66CE                   FE_K[.7]                ;       SETUP SHIFT AMOUNT FOR UNPACK FRAC <H>

                                                        ;-----------------------;
                                                        ID[T3]_D,               ;       SAVE SRC IMMED <L>
                                                        D_LC(FRAC),             ;       UNPACK SRC0 <H>
                                                        Q_0,                    ;       SETUP FOR FRAC <H>
                                                        SS_ALU15,               ;       SS GETS SRC SIGN
                                                        CHK.FLT.OPR,            ;       CHK RSV OPD
U 06CE, 0910,C038,CDF9,3C00,0870,0720                   SET.CC(INST)            ;       SET COND CODES IN CASE DEPEND SOLELY ON SRC

                                                        ;-----------------------;
                                                        D_DAL.SC, SD_SS,        ;       SHIFT RIGHT BY 7, SAVE SS IN SD
                                                        Q_ID[T3],               ;       GET SRC IMMED <L>
                                                        SC_FE,                  ;       SC GETS 7
U 0720, 0D10,0038,CDF4,2EF8,0081,0731                   R[R15]_LC               ;       SAVE SRC0 <H>
```

```
                                        ;10910          ;----------------------;
                                        ;10911          D_DAL.SC, ALU_K[ZERO],  ;  GET SRC FRAC <H>
                                        ;10912          FE_R[R15](EXP),         ;  FE GETS SRC EXP
U 0731, 0D18,0038,1981,FA78,0118,6736   ;10913          CLR.UBCC, SS_ALU15      ;  SET BRANCH COND CODE FOR SRC EXP, CLR SS
                                        ;10914
                                        ;10915          ;----------------------;
                                        ;10916          ID[T0]_D,               ;  SAVE SRC FRAC <H>
                                        ;10917          LC_RC[T1], SS_SD,       ;  LATCH DSTO <H>, RESTORE SS TO OLD VALUE
                                        ;10918          D_Q, Q_0,               ;  SETUP FOR SHIFTING SRC FRAC <L>
U 0736, 0C00,123C,C1FA,3D08,0000,02DA   ;10919          EALU.Z?                 ;  IS SRC EXP = 0 ? (**NOTE SS IS CLEAR!)
                                        ;10920
                                        ;10921   =*01*  ;0---------------------;  NO: SRC .NE. 0 (EALU => 0) (SS=0)
                                        ;10922          SC_LC(EXP),             ;  SC GETS DST EXP
                                        ;10923          SGN/ADD.SUB,            ;  SS GETS FLAG OF +/- FOR ADDD,SUBD
                                        ;10924                                  ;       AND RESULT SIGN FOR MULD, -SIGN FOR DIVD
                                        ;10925                                  ;  SD GETS DST SIGN
                                        ;10926          CHK.FLT.OPR,            ;  CK IF RSV OPD
                                        ;10927          D_DAL.SC,               ;  D GETS SRC FRAC <L>
                                        ;10928          FE_K[.10],              ;  SETUP SHIFT AMOUNT
U 02DA, 0D10,0938,6586,F800,0987,64D8   ;10929          IR2-1?,J/SRCL           ;  GOTO SAVE SRC FRAC <L> AND CK FOR +, -, *, /
                                        ;10930
                                        ;10931          ;1---------------------;  YES:  SRC = 0
                                        ;10932          D_LC,                   ;  D GETS DSTO <H>
                                        ;10933          SC_LC(EXP),             ;  SC GETS DST EXP
                                        ;10934          CHK.FLT.OPR,            ;  CK IF RSV OPD
U 02DE, 0810,C038,0180,F800,08F3,073D   ;10935          SET.CC(INST)            ;  SET COND CODES
                                        ;10936   =;END
                                        ;10937          ;----------------------;
                                        ;10938          Q_RC[T6],               ;  GET DSTO <L>
U 073D, 0010,1438,01C0,F930,0000,0361   ;10939          SC.GT.0?                ;  DST EXP = 0?
                                        ;10940
                                        ;10941   =*01   ;0---------------------;
                                        ;10942          RC[T1]_K[ZERO],         ;  SRC = 0, DST = 0
                                        ;10943          D_0, Q_0,               ;  RESULT IS 0
U 0361, 0F18,003A,19F8,F988,0000,0001   ;10944          RETURN                  ;  RETURN ADR .OR. 1 FOR SRC = DST = 0
                                        ;10945
                                        ;10946          ;1---------------------;  SRC = 0, DST.NE.0
                                        ;10947          RC[T1]_Q,               ;  RC1 GETS DSTO <L>
                                        ;10948          Q_0,                    ;  CLR Q
U 0363, 0001,203E,01F8,F988,0000,0001   ;10949          RETURN1                 ;  RETURN ADR .OR. 1 FOR SRC = 0, DST.NE.0
                                        ;10950   =;END
```

```
                                          ;10951  =00
                                          ;10952  SRCL:    ;00---------------------;  ADDD
                                          ;10953           FE_NABS(SC-LA(EXP)),    ;  FE GETS NABS(DST(EXP) - SRC(EXP))
                                          ;10954           SC_FE,                  ;  SC GETS 16.
                                          ;10955           CLR.UBCC,               ;  SET BRANCH COND CODES
                                          ;10956           ID[T2]_D,               ;  SAVE UNPACKED SRC FRAC <L>
                                          ;10957           D_RC[T6], Q_RC[T6],     ;  GET DSTO <L> FOR SWAP WORD
U 04D8, 0810,1438,C9C0,3D30,0199,E371     ;10958           SC.GT.0?,J/DSTTST       ;  DST EXP = 0?
                                          ;10959
                                          ;10960           ;01---------------------;  SUBD
                                          ;10961           FE_NABS(SC-LA(EXP)),    ;  FE GETS NABS(DST(EXP) - SRC(EXP))
                                          ;10962           SC_FE,                  ;  SC GETS 16.
                                          ;10963           CLR.UBCC,               ;  SET BRANCH COND CODES
                                          ;10964           ID[T2]_D,               ;  SAVE UNPACKED SRC FRAC <L>
                                          ;10965           D_RC[T6], Q_RC[T6],     ;  GET DSTO <L> FOR SWAP WORD
U 04D9, 0810,1438,C9C0,3D30,0199,E371     ;10966           SC.GT.0?,J/DSTTST       ;  DST EXP = 0?
                                          ;10967
                                          ;10968           ;10---------------------;  MULD
                                          ;10969           FE_SC+LA(EXP),          ;  FE GETS DST(EXP) + SRC(EXP)
                                          ;10970           SC_FE,                  ;  SC GETS 16.
                                          ;10971           CLR.UBCC,               ;  SET BRANCH COND CODES
                                          ;10972           ID[T2]_D,               ;  SAVE UNPACKED SRC FRAC <L>
                                          ;10973           D_RC[T6], Q_RC[T6],     ;  GET DSTO <L> FOR SWAP WORD
U 04DA, 0810,1438,C9C0,3D30,0199,8371     ;10974           SC.GT.0?,J/DSTTST       ;  DST EXP = 0?
                                          ;10975
                                          ;10976           ;11---------------------;  DIVD
                                          ;10977           FE_SC-LA(EXP),          ;  FE GETS (DST(EXP) - SRC(EXP))
                                          ;10978           SC_FE,                  ;  SC GETS 16.
                                          ;10979           CLR.UBCC,               ;  SET BRANCH COND CODES
                                          ;10980           ID[T2]_D,               ;  SAVE UNPACKED SRC FRAC <L>
                                          ;10981           D_RC[T6], Q_RC[T6],     ;  GET DSTO <L> FOR SWAP WORD
U 04DB, 0810,1438,C9C0,3D30,0199,A371     ;10982           SC.GT.0?                ;  DST EXP = 0?
                                          ;10983  =;END
                                          ;10984  =*01
                                          ;10985  DSTTST: ;0-----------------------;
                                          ;10986          D_LA,                    ;    DST = 0, SRC.NE.0, D GETS SRC0 <H>
                                          ;10987          LC_RC[T3],               ;    LATCH SRC0 <L>
U 0371, 0800,003E,0180,F918,0000,0002     ;10988          RETURN2                  ;    RETURN ADR.OR.2 FOR DST = 0, SRC.NE.0
                                          ;10989
                                          ;10990          :1-----------------------;    DST.NE.0, SRC.NE.0
                                          ;10991          D_DAL.SC,                ;    SWAP WORD DST FRAC <L>
                                          ;10992          Q_RC[T1](FRAC),          ;    UNPACK DST0 <H>
U 0373, 0D10,0038,B9C8,F908,0084,6755     ;10993          SC_K[.19]                ;    SHIFT AMOUNT SET TO 25.
                                          ;10994  =;END
                                          ;10995          ;------------------------;
                                          ;10996          D_DAL.SC,                ;    SHIFT DST FRAC
                                          ;10997          Q_D,                     ;    GET FRAC LOWER PART
U 0755, 0D00,003C,5DE0,F800,0084,6756     ;10998          SC_K[.7]                 ;    SHIFT AMOUNT SET TO 7
                                          ;10999
                                          ;11000          ;------------------------;
U 0756, 0D00,003C,0180,F800,0000,0759     ;11001          D_DAL.SC                 ;    GET DST FRAC <H>
                                          ;11002
                                          ;11003          ;------------------------;
                                          ;11004          ID[T1]_D, RC[T5]_D,      ;    SAVE DST FRAC <H>
U 0759, 0C01,003C,C5F8,3DA8,0000,0763     ;11005          D_Q, Q_0                 ;    SETUP FOR FRAC <L>
                                          ;11006
                                          ;11007          ;------------------------;
                                          ;11008          D_DAL.SC,                ;    D GETS DST FRAC <L>
                                          ;11009          Q_ID[T2],                ;    Q GETS SRC FRAC <L>
                                          ;11010          SC_RC[T1](EXP),          ;    SC GETS DST EXP
U 0763, 0D10,003A,C9F0,2D08,0083,0003     ;11011          RETURN3                  ;    RETURN ADR .OR. 3 FOR DST.NE.0, SRC.NE.0
```

```
                                         .TOC     "       F & D floating point   : PACK DOUBLE RESULT"

                                         ;ROUTINE TO PACK DOUBLE FLOATING POINT RESULT.
                                         ;ENTRY:          AT "PACKD"
                                         ;INPUTS:         D = NORMALIZED FRACTION<H>, UNROUNDED
                                         ;                Q = NORMALIZED FRACTION<L>, ROUNDED
                                         ;                C31 = CARRY FROM ROUNDING Q
                                         ;                SC = BIASED EXPONENT
                                         ;                SD = SIGN
                                         ;                FE = 18 (HEX)
                                         ;                STATE<0> = 1 IF CALLED FROM POLYD, ELSE 0
                                         ;                STATE<3:1> = 0
                                         ;OUTPUTS: (IF NOT CALLED FROM POLYD)
                                         ;                D = RESULT<H>
                                         ;                Q = RC[T1] = RESULT<L>
                                         ;                CONDITION CODES SET
                                         ;                TRAP CODE SET IN ID[CES] IF UNDERFLOW OR OVERFLOW OCCURRED
                                         ;OUTPUTS: (IF CALLED FROM POLYD)
                                         ;                Q = RC[T1] = RESULT<H>
                                         ;                D = RESULT<L>
                                         ;                SC = FE = EXPONENT
                                         ;TEMPORARIES:    LC
                                         ;RETURNS:        RETURNS @ 10 IF NOT CALLED FROM POLYD
                                         ;                RETURNS @ 10 IF CALLED FROM POLYD AND EXPONENT = 0
                                         ;                RETURNS @ 12 OR 13 IF CALLED FROM POLYD AND 1 < EXPONENT < 100
                                         ;                RETURNS @ 15 IF CALLED FROM POLYD AND EXPONENT < 0
                                         ;                RETURNS @ 17 IF CALLED FROM POLYD AND EXPONENT > FF

                                         =0*
                                         PACKD.0: ;0----------------------;
                                                  D_DAL.SC, SC_FE,          ;  D GETS FRAC <L>, SC GETS BACK EXP
                                                  EALU_K[.1], CLK.UBCC,     ;  CLEAR EALU CC'S
U 03EC, 0D00,173C,0580,F800,0095,63E4             STATE0?, J/PACKD.2        ;  CHECK FOR POLYD AND GO SWAP FRAC<L>

                                                  ;1----------------------;
                                                  D_D+K[.1],                ;  FRAC <H> INCREMENTED BY 1
                                                  SC_FE,FE_SC,              ;  SC GETS EXP, FE GETS 24.
U 03EE, 0819,0014,0580,F800,0191,0768             CLK.UBCC                  ;  CLOCK IN CARRY IF ANY
                                         =;END
                                                  ;-----------------------;
                                                  ALU_0+K[.1],CLK.UBCC,     ;  CLR C31
U 0768, 001B,0314,0580,F800,0010,03FC             C31?                      ;  HAVE TO INCREMNET EXP BY 1?
                                         =0*
                                         PACKD:   ;0----------------------;  NO, PACK RESULT <H>
                                                  SC_FE,FE_SC,              ;  SC GETS 24. AND FE SAVES EXP
                                                  RC[T1]_PACK.FP,           ;  RC1 GETS PACKED RESULT <H>
                                                  SET.CC(INST),             ;  SET COND CODES
                                                  C31?,                     ;  HAVE TO INCREMENT FRAC <H>?
U 03FC, 0008,C338,0180,F988,01F1,03EC             J/PACKD.0                 ;  GOTO PACK FRAC <L> OR INCREMENT FRAC <H>

                                                  ;1----------------------;  YES, INCREMNET EXP BY 1
U 03FE, 0000,003C,0180,F800,0080,C3FC             SC_SC+1,J/PACKD           ;
                                         =;END
```

```
                                      =***0
                                      PACKD.2: ;0---------------------;  NOT POLYD
                                               SC_K[.10].ALU,         :  SC GETS 16. FOR WORD SWAP FOR FRAC <L>
                                               FE_SC,                 :  SAVE EXP FOR SETTING COND CODES LATTER
                                               Q_D,                   :  READY FOR SWAP WORD FOR FRAC <L>
U 03E4, 0018,1438,65E0,F800,0182,0661          SC?,J/PACKD.4          :  EXP SHOWS 0  NON-0, UNDERFLOW, OVERFLOW

                                               ;1---------------------;  POLYD
                                               SC_K[.10].ALU,         :  SC GETS 16. FOR WORD SWAP FOR FRAC <L>
                                               FE_SC,                 :  SAVE EXP FOR SETTING COND CODES LATTER
                                               Q_D,                   :  READY FOR SWAP WORD FOR FRAC <L>
U 03E5, 0018,0038,65E0,F800,0182,0775          J/PACKD.8              :
                                      =;END
                                      =0001
                                      PACKD.4:;00---------------------;  0 EXP: RESULT UNDERFLOW
                                               D_0,Q_0,EALU_K[ZERO],  :  SET RESULT <H,L> TO ZEROS
                                               RC[T1]_K[ZERO],        :  RESULT <L> SET TO 0
                                               SET.CC(INST),          :  SET COND CODES
U 0661, 0F18,C039,19F8,F988,0074,6E09          CALL,J/UNDRFL          :  SET CES FOR UNDERFLOW

                                               ;01--------------------;  01 TO FF EXP: OK
                                               D_DAL.SC,              :  SWAP WORD
                                               Q_RC[T1],              :  Q GETS RESULT <H>
U 0663, 0D10,0038,01C0,F908,0000,0771          J/PACKD.6              :

                                               ;10--------------------;  NEG EXP: UNDERFLOW
                                               D_0,Q_0,EALU_K[ZERO],  :  SET RESULT <H,L> TO ZEROS
                                               RC[T1]_K[ZERO],        :  RESULT <L> SET TO 0
                                               SET.CC(INST),          :  SET COND CODES
U 0665, 0F18,C039,19F8,F988,0074,6E09          CALL,J/UNDRFL          :  SET CES FOR UNDERFLOW

                                      PACKD.OV:;11--------------------;  > 31 EXP: OVERFLOW
                                               EALU_FE,               :  PASS EXP FOR COND CODES DETECTION
                                               D_K[.8000],            :  RESULT SET TO -0
                                               SET.CC(INST),          :  SET COND CODES
U 0667, 0818,C039,4580,F800,0070,6E03          CALL,J/OVFL            :  SET CES FOR OVERFLOW

                                      =1111    ;----------------------;  RETURN FROM CES SETTING
                                               RC[T1]_K[ZERO],Q_0,    :  RESULT <L> SET TO 0
U 066F, 0018,003A,19F8,F988,0000,0010          RETURN10               :  RETURN TO INSTR CALLED FROM FORK ENTRIES
                                      =;END

                                      PACKD.6:;-----------------------;
                                               RC[T1]_D, Q_D,         :  RC1, Q HAVE RESULT <L>
                                               D_Q,                   :  D GETS RESULT <H>
U 0771, 0C01,003E,01E0,F988,0000,0010          RETURN10               :  RETURN TO INSTR CALLED FROM FORK ENTRIES

                                      PACKD.8:;-----------------------;  POLYD
                                               D_DAL.SC,SC_FE,        :  SWAP WORD, GET BACK EXP
U 0775, 0D10,0038,01C0,F908,0081,0776          Q_RC[T1]               :  Q GETS RESULT <H>
                                      =;END
                                               ;----------------------;
U 0776, 0000,143E,0180,F800,0000,0010          SC?,RETURN10           :  RETURN TO POLYD
```

```
                                         ;11118  .TOC    "       F & D floating point   : ADDD, SUBD"
                                         ;11119
                                         ;11120  ;ADDD2/SUBD2,  SHORT LITERAL (SL) - REGISTER (R)
                                         ;11121  ;ENTER HERE FROM IRD STATE AT A.FORK WITH SL IN Q, R IN LA.
                                         ;11122
                                         ;11123  042:     ;-------------------------;
                                         ;11124           STATE_K[ZERO],            ;   CLR POLYD FLAG
U 0042, 0001,203C,19F8,F980,1404,6778    ;11125           RC[T0]_Q, Q_0             ;   SRCO <RC0, Q> GETS <SL, R>
                                         ;11126
                                         ;11127           ;-------------------------;
U 0778, 0000,003C,0180,F988,0000,077B    ;11128           RC[T1]_LA                 ;   RC1 GETS DSTO <H>
                                         ;11129
                                         ;11130  ADDD.A:  ;-------------------------;
U 077B, 0800,003C,0180,F870,0000,0225    ;11131           D_R(SP1+1)                ;   D GETS DSTO <L>
                                         ;11132
                                         ;11133  =0****
                                         ;11134  ADDD.R:  ;0------------------------;
                                         ;11135           RC[T6]_D,                 ;   SAVE DSTO <L>
                                         ;11136           D_Q,                      ;   D GETS SRCO <L>
                                         ;11137           SC_K[.10],                ;   SC GETS 16. FOR SWAP WORD OF FRAC <L>
U 0225, 0C01,003D,6580,F980,0084,6580    ;11138           CALL,J/ADDD.S             ;   CALL ADDD/SUBD SUBROUTINE
                                         ;11139
                                         ;11140           ;1------------------------;   RETURN WITH RESULT IN <D, RC1>
U 0235, 0001,003C,0180,F8C0,0000,0782    ;11141           R(SP1)_D                  ;   STORE RESULT <H>
                                         ;11142  =;END
                                         ;11143
                                         ;11144           ;-------------------------;
U 0782, 0000,003C,0180,F908,0000,0786    ;11145           LC_RC[T1]                 ;   GET RESULT <L>
                                         ;11146
                                         ;11147           ;-------------------------;
                                         ;11148           R(SP1+1)_LC,              ;   STORE RESULT <L>
                                         ;11149           CLR.IB0-1,PC_PC+2,        ;   CLR IB BYTES 0,1
U 0786, 4010,0038,0180,F8F5,4000,0062    ;11150           J/IRD                     ;   GOTO NEXT INST
                                         ;11151
                                         ;11152
                                         ;11153  ;ADDD2/SUBD2,  R-R
                                         ;11154  ;ENTER HERE FROM IRD STATE AT A.FORK WITH LB, LA HAVE SRC <H>, DST<H>
                                         ;11155
                                         ;11156  046:     ;-------------------------;
                                         ;11157           STATE_K[ZERO],            ;   CLR POLYD FLAG
U 0046, 000C,0038,1980,F980,1404,6789    ;11158           RC[T0]_LB                 ;   RCO GETS SRCO <H>
                                         ;11159
                                         ;11160           ;-------------------------;
U 0789, 0000,003C,0180,F988,0000,078A    ;11161           RC[T1]_LA                 ;   RC1 GETS DSTO <H>
                                         ;11162
                                         ;11163           ;-------------------------;
                                         ;11164           Q_R(PRN+1),               ;   Q GETS SRCO <L>
U 078A, 0000,003C,01C0,F860,0000,077B    ;11165           J/ADDD.A                  ;   GOTO SAME FLOW AS SL-R
```

```
                                     ;ADDD3/SUBD3, *-SL-R
                                     ;ENTER HERE AT B.FORK WITH SRCO IN <RCO, D>, SL IN Q.
                                     ;LEAVE HERE WITH SRCO IN <RCO, Q>, DSTO IN <RC1, D>

                                     242:      ;-----------------------;
                                               STATE_K[ZERO],          ;    CLR POLYD FLAG
                                               RC[T1]_Q,               ;    RC1 GETS DSTO <H>
                                               Q_D, D_0,               ;    Q GETS SRCO <L>, D GETS DSTO <L> WHICH IS 0
U 0242, 0F01,203C,19E0,F988,1404,6225          J7ADDD.R                ;

                                     ;ADDD3/SUB3, *-R-R

                                     246:      ;-----------------------;
                                               STATE_K[ZERO],          ;    CLR POLYD FLAG
                                               RC[T1]_LA,              ;    RC1 GETS DSTO <H>
U 0246, 0000,003C,19E0,F938,1404,678E          Q_D                     ;    Q GETS SRCO <L>

                                               ;-----------------------;
                                               D_R(PRN+1),             ;    GET DSTO <L>
U 078E, 0800,003C,0180,F860,0000,0225          J7ADDD.R                ;

                                     ;ADDD2/SUBD2, *-R
                                     ;ENTER HERE AT B.FORK WITH SRCO IN <RCO, D>, R IN LA, LB.
                                     ;LEAVE HERE WITH SRCO IN <RCO, Q>, DSTO IN <RC1, D> TO GOTO SAME FLOW
                                     ;AS SL-R, R-R.

                                     22E:      ;-----------------------;
                                               STATE_K[ZERO],          ;    CLR POLYD FLAG
                                               RC[T1]_LA,              ;    RC1 GETS DSTO <H>
U 022E, 0000,003C,19E0,F988,1404,6791          Q_D                     ;    Q GETS DSTO <L>

                                               ;-----------------------;
U 0791, 0800,003C,0180,F860,0000,00EC          D_R(PRN+1)              ;    D GETS DSTO <L>

                                     =0110*    ;00---------------------;
                                               RC[T6]_D,               ;    SAVE DSTO <L>
                                               D_Q,                    ;    D GETS SRCO <L>
                                               SC_K[.10],              ;    SC GETS 16. FOR SWAP WORD OF FRAC <L>
U 00EC, 0C01,003D,6580,F9B0,0084,6580          CALL.J/ADDD.S           ;    CALL ADDD/SUBD SUBROUTINE

                                     =1110*
                                     ADDD.M:   ;10---------------------;    RETURN WITH RESULT IN <D, RC1>
U 00FC, 0001,003C,0180,F8D8,0000,0796          R(PRN)_D,J/ADDD.P       ;    STORE RESULT <H>

                                               ;11---------------------;    RETURN WITH RESULT IN <D, RC1>
U 00FE, 0001,003C,0180,F8D8,0020,0796          R(PRN)_D,SET.V          ;    RESET PSL<V> FOR DIVD
                                     =;END

                                     ADDD.P:   ;-----------------------;
U 0796, 0000,003C,0180,F908,0000,079A          LC_RC[T1]               ;    GET RESULT <L>

                                               ;-----------------------;
                                               R(PRN+1)_LC,            ;    STORE RESULT <L>
                                               CLR.IB.OPC,PC_PC+1,     ;    CLR IB BYTE 0
U 079A, C010,0038,0180,F8E4,4000,0062          J/IRD                   ;    GOTO NEXT INST
```

```
                                         ;11221  ;DOUBLE FLOATING POINT ADDD AND SUBD *-*  OR  *-*-*
                                         ;11222  ;ENTER AT C.FORK WITH SRC OPD IN <RC0, Q>, DST OPD IN <RC1, D>.
                                         ;11223  ;ALWAYS YIELDS NORMALIZED AND ROUNDED RESULT.
                                         ;11224
                                         ;11225  38A:
                                         ;11226  ADDD:    ;------------------------;
U 038A, 0000,003C,1980,F800,1404,62A4    ;11227           STATE_K[ZERO]           ;   CLR POLYD FLAG
                                         ;11228
                                         ;11229  =0****   ;0-----------------------;
                                         ;11230           RC[T6]_D,               ;   SAVE DSTO <L>
                                         ;11231           D_Q,                    ;   D GETS SRCO <L>
                                         ;11232           SC_K[.10],              ;   SC GETS 16. FOR SWAP WORD OF FRAC <L>
U 02A4, 0C01,003D,6580,F9B0,0084,6580    ;11233           CALL, J/ADDD.S          ;   CALL ADDD/SUBD SUBRT
                                         ;11234
                                         ;11235           ;1-----------------------;
U 02B4, F000,003F,01F0,F847,0000,0300    ;11236           WRITE.DEST,J/WRD        ;   WRITE RESULT
                                         ;11237  =;END
```

```
                                          ;       COMMON DOUBLE FLOATING ADD/SUB ROUTINE
                                          ;
                                          ;INPUTS:         RC[T0] = SRC<H>
                                          ;                D = Q = SRC<L>
                                          ;                RC[T1] = DST<H>
                                          ;                RC[T6] = DST<L>
                                          ;                SC = 10 (HEX)
                                          ;                STATE = 0
                                          ;                IR<1> = 0 IF ADDD, 1 IF SUBD
                                          ;
                                          ;OUTPUTS:        D = PACKED ROUNDED SUM<H>
                                          ;                RC[T1] = PACKED ROUNDED SUM<L>
                                          ;                CONDITION CODES AND ID[CES] SET ON SUM
                                          ;
                                          ;TEMPORARIES:    R[R15], RC[T2], RC[T3], RC[T5]
                                          ;                ID[T0], ID[T1], ID[T2], ID[T3],
                                          ;                SS, SD, Q, FE, LA, LB, LC
                                          ;
                                          ;RETURNS:        RETURNS @ 10
                                          ;
                                          =00
                                          ADDD.S: ;00---------------------;
                                                  RC[T3]_D,               ;    SAVE SRCO <L>
                                                  D_DAL.SC,               ;    SWAP WORD OF SRCO <L>
                                                  SC_K[.FFF9],            ;    SETUP SHIFT AMOUNT -7
U 0580, 0D01,003D,BD80,F998,0084,66CA             CALL,J/UNPACK           ;    CALL UNPACK DOUBLE FLOATING PT OPERANDS ROUTINE

                                                  ;01---------------------;    RETURN1, SRC = 0, DST MAY BE 0
                                                  ALU_D,                  ;    SET COND CODES AS DST (UNPACK ASSURES CLEAN 0)
                                                  SET.CC(INST),           ;    [illegible]IR DEP COND CODES
U 0581, 0001,C03E,0180,F800,0070,0010             RETURN10                ;    GOTO WRITE DEST

                                                  ;10---------------------;    RETURN2, DST = 0, SRC.NE.0
                                                  RC[T1]_LC,              ;    RC1 GETS SRCO <L>
U 0582, 0010,0938,0180,F988,0000,05B2             IR1?, J/ADDD.8          ;    ADDD OR SUBD?

                                                  ;11---------------------;    RETURN3, SRC.NE.0, DST.NE.0
                                          ADDD.6: LC_RC[T1],              ;    LATCH UP PACKED HI DEST IN LC
U 0583, 0000,123C,0180,F908,0000,0182             EALU?,J/ADDD.10         ;    BEN ON EALU<N,Z>, SS
                                          =;END

                                          =10
                                          ADDD.8: ;0----------------------;    CONSTRAINED BLOCK FOR IR1?
                                                  ALU_D,SET.CC(INST),     ;    ADDD: RESULT = SRC,  SET COND CODES
U 05B2, 0001,C03E,0180,F800,0070,0010             RETURN10                ;    GOTO WRITE DEST

                                                  ;1----------------------;    SUBD: RESULT = -SRC
                                                  D_D.XOR.K[.8000],       ;    RESULT = -SRC
                                                  SET.CC(INST),           ;    SET COND CODES
U 05B3, 0819,C022,4580,F800,0070,0010             RETURN10                ;    GOTO WRITE DEST
                                          =;END
```

```
                                        ;11289  ;        MAIN BRANCHING POINT OF ADDD - ALSO USED BY POLYD AND ACBD
                                        ;11290  ;        AT THIS POINT, MACHINE STATE IS AS FOLLOWS
                                        ;11291  ;        D=DST FRAC<L>, Q&ID[T2]=SRC FRAC<L>, RC[T5]&ID[T1]=DST FRAC<H>,
                                        ;11292  ;        ID[T0]=SRC FRAC<H>, SC=DST EXP, FE=NABS(SRC EXP - DST EXP),
                                        ;11293  ;        SD=DST SGN, SS=DST SGN .XOR. SRC SGN .XOR. SUBD,
                                        ;11294  ;        RC[T0]=ORIG SRC<H>, LC&RC[T1]=ORIG DST<H>, RC[T6]=ORIG DST<L>,
                                        ;11295  ;        STATE<0> = POLYD FLAG, STATE<3:1> = 0.
                                        ;11296
                                        ;11297  =000010
                                        ;11298  ADDD.10:;0010-------------------;   DST EXP > SRC EXP, ADD
                                        ;11299          R[R15]_D,                ;   SAVE DST FRAC <L>
                                        ;11300          D_Q,                     ;   SETUP FOR ALIGN SRC FRAC
                                        ;11301          SC_FE,                   ;   GET SHIFT AMOUNT
                                        ;11302          Q_ID[T0],                ;   GET SRC FRAC <H>
U 0182, 0C01,003D,C1F0,2EF8,0081,04AE   ;11303          CALL[ALNPOS]             ;   GO ALIGN SRC
                                        ;11304
                                        ;11305          ;0011-------------------;   DST EXP > SRC EXP, SUB
                                        ;11306          R[R15]_D,                ;   SAVE DST FRAC <L>
                                        ;11307          D_Q,                     ;   SETUP FOR ALIGN SRC FRAC
                                        ;11308          SC_FE,                   ;   GET SHIFT AMOUNT
                                        ;11309          Q_ID[T0],                ;   GET SRC FRAC <H>
U 0183, 0C01,003D,C1F0,2EF8,0081,080E   ;11310          CALL[ALNNEG]             ;   TWO'S COMPLEMENT AND ALIGN SRC
                                        ;11311
                                        ;11312  ADDD.14:;0110-------------------;   DST EXP = SRC EXP, ADD
                                        ;11313          R[R15]_D+Q,              ;   ADD FRAC <L>'S
                                        ;11314          Q_ID[T1],                ;   Q GETS DST FRAC <H>
                                        ;11315          CLK.UBCC,                ;   CLOCK IN CARRY IF ANY
                                        ;11316          SC_SC+1,                 ;   ADD 1 TO RESULT EXP TO OFFSET SHF.VAL COUNTING
U 0186, 001D,0014,C5F0,2EF8,0090,C7BD   ;11317          J/ADDD.26                ;
                                        ;11318
                                        ;11319  ADDD.16:;0111-------------------;   DST EXP = SRC EXP, SUB
                                        ;11320          R[R15]_D-Q,              ;   SUB FRAC <L>'S
                                        ;11321          Q_ID[T1],                ;   Q GETS DST FRAC <H>
                                        ;11322          CLK.UBCC,                ;   CLOCK IN CARRY IF ANY
                                        ;11323          SC_SC+1,                 ;   ADD 1 TO RESULT EXP TO OFFSET SHF.VAL COUNTING
U 0187, 001D,0000,C5F0,2EF8,0090,C7DA   ;11324          J/ADDD.30                ;
                                        ;11325
                                        ;11326          ;1010-------------------;   DST EXP < SRC EXP, ADD
                                        ;11327          SC_FE,                   ;   SC GETS SHIFT AMOUNT FOR ALIGNMENT
                                        ;11328          Q_ID[T1], LC_RC[T0],     ;   Q GETS DST FRAC <H>, LC GETS SRC EXP
U 018A, 0000,003D,C5F0,2D00,0081,04AE   ;11329          CALL[ALNPOS]             ;   GO ALIGN DST
                                        ;11330
                                        ;11331          ;1011-------------------;   DST EXP < SRC EXP, SUB
                                        ;11332          SC_FE,                   ;   SC GETS SHIFT AMOUNT FOR ALIGNMENT
                                        ;11333          Q_ID[T1], LC_RC[T0],     ;   Q GETS DST FRAC <H>, LC GETS SRC EXP
                                        ;11334          SD_NOT.SD,               ;   RESULT SIGN IS SRC SIGN
U 018B, 0000,003D,C5F3,2D00,0081,080E   ;11335          CALL[ALNNEG]             ;   TWO'S COMP AND ALIGN DST
```

```
                                          ;11336  ;        ADDD - RETURNS FROM ALIGNMENT ROUTINE
                                          ;11337
                                          ;11338  =010010 ;----------------------;  DST EXP > SRC EXP, ADD, DIFF<32
                                          ;11339          SC_FE, FE_SC, D_Q, Q_0, ;  SC GETS SHIFT AMT, FE GETS DST EXP,
U 0192, 0C00,003C,C9F8,3C00,0181,079E     ;11340          ID[T2]_D, J/ADDD.21     ;  PREPARE TO SHIFT HIGH-ORDER SRC
                                          ;11341
                                          ;11342          ;----------------------;  DST EXP > SRC EXP, SUB, DIFF<32
                                          ;11343          SC_FE, FE_SC, D_Q,      ;  SC=SHIFT AMT, FE=DST EXP,
                                          ;11344          ALU_-1, Q_ALU,          ;  SIGN-EXTENSION OF FRACT TO Q, PREPARE
U 0193, 0C03,0028,C9C0,3C00,0181,079E     ;11345          ID[T2]_D, J/ADDD.21     ;  TO SHIFT HIGH-ORDER NEGATED SRC FRAC
                                          ;11346
                                          ;11347  =011010 ;----------------------;  DST EXP < SRC EXP, ADD, DIFF<32
                                          ;11348          SC_FE, FE_SC, D_Q, Q_0, ;  SC=SHIFT AMT, FE=SRC EXP, PREPARE
U 019A, 0C01,003C,01F8,F990,0181,07B9     ;11349          RC[T2]_D, J/ADDD.24     ;  TO SHIFT HIGH-ORDER DST FRAC
                                          ;11350
                                          ;11351          ;----------------------;  DST EXP < SRC EXP, SUB, DIFF<32
                                          ;11352          SC_FE, FE_SC, D_Q,      ;  SC=SHIFT AMT, FE=SRC EXP, PREPARE
U 019B, 0C01,003C,0180,F990,0181,07B1     ;11353          RC[T2]_D, J/ADDD.23     ;  TO SHIFT HIGH-ORDER NEGATED DST FRAC
                                          ;11354
                                          ;11355  =100011 ;----------------------;  DST EXP > SRC EXP, ADD/SUB, 32<=DIFF<64
                                          ;11356          ID[T2]_D, Q_D, D_Q,     ;  SAVE LOW ALIGNED SRC, D GETS SIGN EXT
U 01A3, 0C00,003C,C9E0,3C00,0000,07A9     ;11357          J/ADDD.22
                                          ;11358
                                          ;11359  =101011 ;----------------------;  DST EXP < SRC EXP, ADD/SUB, 32<=DIFF<64
                                          ;11360          RC[T2]_D, Q_ID[T2],     ;  SAVE LOW ALIGNED DST, Q GETS LOW SRC,
U 01AB, 0C01,003C,C9F0,2D90,0000,07BA     ;11361          D_Q, J/ADDD.25          ;  D GETS 32*DST FRAC SIGN
                                          ;11362
                                          ;11363  =110011 ;----------------------;  DST EXP > SRC EXP, ADD/SUB, DIFF>63
                                          ;11364          Q_RC[T6], SC_D(EXP)(A), ;  D HAS ORIG DST<H>, SET Q TO ORIG DST<L>
U 01B3, 0011,1738,01C0,F930,0088,6428     ;11365          STATE0?, J/ADDD.20      ;  CHECK TO SEE IF WE WERE CALLED FROM POLYD
                                          ;11366
                                          ;11367  =111011 ;----------------------;  DST EXP < SRC EXP, ADD/SUB, DIFF>63
                                          ;11368          D_Q, Q_ID[T2], FE_SC+1, ;  ANSWER IS +/- SRC - RECONSTRUCT FROM FRAC&EXP
U 01BB, 0C00,003C,C9F0,2C00,0100,C67B     ;11369          J/ADDD.31               ;  TO GET SIGN RIGHT (FROM SD) AND BECAUSE
                                          ;11370                                  ;  POLYD DOESN'T HAVE A PACKED VERSION OF SRC.
                                          ;11371  =;END
```

```
                                            =***0
                                            ADDD.20:;0---------------------;   HERE WHEN SRC=0 W.R.T. DST AND NOT POLYD
U 0428, 0001,203E,0180,F988,0000,0010               RC[T1]_Q, RETURN10     :   STORE DST<L> IN RESULT<L> AND EXIT

                                            ; **********************************************
                                            ; * Patch no. 031, PCS 0428 trapped to WCS 1163 *
                                            ; **********************************************

                                                    ;1---------------------;   HERE WHEN SRC=0 W.R.T. DST AND INST IS POLYD
U 0429, 0C00,143E,01E0,F800,0000,0010               D_Q, Q_D, SC?, RETURN10 :  PUT RESULT IN <Q,D> & RETURN (12)
                                            =;END
                                            ADDD.21:;----------------------;   HERE WHEN SRC IS TO BE ALIGNED <32 PLACES
                                                    D_DAL.SC,              :   D GETS ALIGNED SRC FRAC <H>
                                                    SC_FE,                 :   SC GETS BACK DST EXP
U 079E, 0D00,003C,C9F0,2C00,0081,07A9               Q_ID[T2]               :   Q GETS DST FRAC <L>

                                            ADDD.22:;----------------------;   HERE WHEN SRC FRACTION IS ALIGNED
                                                    ID[T0]_D,              :   ID[T0] GETS ALIGNED SRC FRAC <H>
                                                    D_R[R15],              :   D GETS DST FRAC <H>
U 07A9, 0800,003C,C180,[illegible]78,0000,0186               J/ADDD.14              :   GOTO ADD FRAC <L>

                                            ADDD.23:;----------------------;   UGLY STATE
U 07B1, 0003,0028,01C0,F800,0000,07B9               ALU_-1, Q_ALU          :   COULDN'T DO THIS AND SAVE DST<L> IN RC[T2]
                                                                           :   IN THE SAME STATE

                                            ADDD.24:;----------------------;   HERE TO COMPLETE DST ALIGN OF <32 BITS
                                                    D_DAL.SC,              :   D GETS ALIGNED DST FRAC <H>
                                                    SC_FE,                 :   SC GETS SRC EXP
U 07B9, 0D00,003C,C9F0,2C00,0081,07BA               Q_ID[T2]               :   Q GETS SRC FRAC <L>

                                            ADDD.25:;----------------------;   HERE WHEN DST FRAC IS ALIGNED
                                                    ID[T1]_D,              :   STORE ALIGNED DST FRAC <H>
                                                    D_RC[T2],              :   D GETS ALIGNED DST FRAC <L>
U 07BA, 0810,0038,C580,3D10,0000,0186               J/ADDD.14              :   GOTO ADD FRAC <L>
```

```
                                          ;COME HERE FROM ADDD.14 AFTER ADDING LOW-ORDER FRACTIONS WHEN REAL OPERATION
                                          ;         IS SUBTRACT WITH UNEQUAL EXPONENTS OR ADD.

                                          ADDD.26:;----------------------;
                                                   D_Q,                  ;     D GETS ALIGNED DST FRAC <H>
                                                   Q_ID[T0],             ;     Q GETS ALIGNED SRC FRAC <H>
                                                   FE_SC,                ;     FE GETS EXP
U 07BD, 0C00,033C,C1F0,2C00,0100,0420              C31?                  ;     HAVE TO ADD 1 MORE FOR FRAC <H>'S?

                                          =0*      ;---------------------;
                                                   D_D+Q,                ;     ADD ALIGNED FRAC <H>'S
                                                   LAB_R[R15], CLK.UBCC, ;     LATCH RESULT FRAC <L>, SET ALU.Z ON FRAC<H>
U 0420, 081D,0014,0180,FA78,0010,07CA              J/ADDD.27             ;

                                                   ;1--------------------;
                                                   D_D+Q+1,              ;     ADD ALIGNED FRAC <H>'S PLUS 1
U 0422, 081D,0010,0180,FA78,0010,07CA              LAB_R[R15], CLK.UBCC  ;     LATCH RESULT FRAC <L>, SET ALU.Z ON FRAC<H>
                                          =;END

                                          ADDD.27:;----------------------;
                                                   Q_LA,                 ;     Q GETS RESULT FRAC <L>
U 07CA, 0000,1B3C,01C0,F800,0000,067B              ALU?, J/ADDD.31       ;     GOTO NORMALIZE (UNLESS FRAC<H>=0)

                                          ;COME HERE FROM ADDD.16 AFTER SUBTRACTING LOW-ORDER FRACTIONS
                                          ;WHEN REAL OPERATION IS SUBTRACT WITH EQUAL EXPONENTS.

                                          ADDD.30:;----------------------;
                                                   D_Q,                  ;     D GETS ALIGNED DST FRAC <H>
                                                   Q_ID[T0],             ;     Q GETS ALIGNED SRC FRAC <H>
U 07DA, 0C00,033C,C1F0,2C00,0000,0474              C31?                  ;     HAVE TO SUB 1 MORE FOR FRAC <H>'S?

                                          =0*      ;0--------------------;     DST FRAC <L> < SRC FRAC <L>
                                                   D_D-Q-1,              ;     SUB ALIGNED FRAC <H>'S PLUS 1
                                                   CLK.UBCC,             ;     CLOCK ALU.N BIT
                                                   LAB_R[R15],           ;     LATCH RESULT FRAC <L>
U 0474, 081D,0008,0180,FA78,0010,07DC              J/ADDD.32             ;

                                                   ;1--------------------;     DST FRAC <L> .GE. SRC FRAC <L>
                                                   D_D-Q,                ;     SUB ALIGNED FRAC <H>'S
                                                   CLK.UBCC,             ;     CLOCK ALU.N BIT
U 0476, 081D,0000,0180,FA78,0010,07DC              LAB_R[R15]            ;     LATCH RESULT FRAC <L>
                                          =;END

                                          ADDD.32:;----------------------;
                                                   FE_SC,                ;     SAVE DST EXP IN FE
                                                   Q_R[R15],             ;     Q GETS DIFF FRAC <L>
U 07DC, 0000,1B3C,01C0,FA78,0100,067A              ALU?                  ;     IS DIFF NEG?

                                          =1010    ;00-------------------;
                                                   SD_NOT.SD,            ;     NEG DIFF FRAC, NEG SIGN
                                                   Q_0-Q,                ;     NEG FRAC <L>
                                                   CLK.UBCC,             ;     CLOCK IN ALU.Z BIT
U 067A, 001F,0000,01C3,F800,0010,07E5              J/ADDD.33             ;
```

```
                                        ;11459  :         ALL ADDS AND SUBTRACTS CONVERGE IN ONE OF THE NEXT TWO STATES.
                                        ;11460  :         EMODD ALSO ENTERS HERE TO NORMALIZE AND PACK ITS FRACTION RESULT.
                                        ;11461
                                        ;11462  =1011
                                        ;11463  ADDD.31:;01---------------------;  RESULT FRAC <H> IS NOT 0
                                        ;11464            SC_SHF.VAL,           ;  POS DIFF, NORMALIZE RESULT
                                        ;11465            D_DAL.NORM,           ;  D GETS NORMALIZED FRAC <H>
U 067B, 0E00,003C,0180,F800,008C,67E8   ;11466            J/ADDD.36             ;
                                        ;11467
                                        ;11468  =1111     ;11---------------------;  RESULT FRAC <H> IS 0
                                        ;11469            SC_SC-K[.20],         ;  ADJUST EXP
                                        ;11470            FE_SC-K[.20],         ;  COUNT THE DUMMY LEADING 0
                                        ;11471            D_0,Q_0,              ;  SET FRAC <H>, CLR FRAC <L>
U 067F, 0C00,003C,75F8,F800,0184,A67B   ;11472            J/ADDD.31             ;
                                        ;11473  =;END
```

```
                                           ;11474  ADDD.33:;---------------------;
                                           ;11475          D_NOT.D,CLK.UBCC,     :   1'S COMP OF FRAC <H>
U 07E5, 0801,0128,0180,F800,0010,042C      ;11476          Z?                    :   IS LOW FRAC ALL ZEROS?
                                           ;11477
                                           ;11478  =0      ;0--------------------;
U 042C, 0000,1B3C,0180,F800,0000,067B      ;11479          ALU?,J/ADDD.31        :   IS DIFF ZERO?
                                           ;11480
                                           ;11481          ;1--------------------;
                                           ;11482          D_D+K[.1],            :   YES, FRAC <L> = 0, MAKE 2'S COMP OF FRAC <H>
U 042D, 0819,0014,0580,F800,0000,067B      ;11483          J/ADDD.31             :
                                           ;11484  =;END
                                           ;11485
                                           ;11486  ADDD.36:;---------------------;   COME HERE WITH FRAC<H> NORMALIZED
                                           ;11487          R[R15]_D,             :   SAVE FRAC <H>
U 07E8, 0C01,003C,01F8,FAF8,0000,07F5      ;11488          D_Q,Q_0               :   SETUP FOR NORMALIZE FRAC <L>
                                           ;11489
                                           ;11490          ;---------------------;
                                           ;11491          D_DAL.SC,             :   NORMALIZE FRAC <L>
                                           ;11492          SC_FE,FE_SC,          :   SC GETS EXP, FE GETS #OF LEADING ZEROS
U 07F5, 0D00,003C,01C0,FA78,0181,07FE      ;11493          Q_R[R15]              :   Q GETS BACK FRAC <H>
                                           ;11494
                                           ;11495          ;---------------------;
                                           ;11496          SC_SC-FE,             :   EXP_EXP - #OF LEADING ZEROS
                                           ;11497          D_Q,Q_0,              :
U 07FE, 0C01,003C,01F8,FAF8,0080,A806      ;11498          R[R15]_D              :
                                           ;11499
                                           ;11500          ;---------------------;
                                           ;11501          LAB_R[R15],           :   LATCH FRAC <L>
                                           ;11502          ALU_LA[OR]D,CLK.UBCC, :   CHECK IF RESULT FRAC'S ARE ZEROS
U 0806, 001C,2030,7D80,FA78,0114,680A      ;11503          FE_K[.18]             :   SET FE TO 24. FOR PACKING DOUBLE RESULT
                                           ;11504
                                           ;11505          ;---------------------;
                                           ;11506          K[.80],               :   PRESET SLOW CONSTANT FOR ROUNDING
U 080A, 0000,013C,4180,F800,0000,0440      ;11507          Z?                    :   RESULT FRAC'S = 0?
                                           ;11508
                                           ;11509  =0      ;0--------------------;   RESULT FRAC'S .NE. 0
                                           ;11510          Q_LA+K[.80],          :   ADD 80(HEX) FOR ROUNDING
                                           ;11511          CLK.UBCC,             :   CLOCK IN FOR CARRY IF ANY
U 0440, 0018,0014,41C0,F800,0010,03FC      ;11512          J/PACKD               :   GOTO PACK DOUBLE FLOATING RESULT
                                           ;11513
                                           ;11514          ;1--------------------;   RESULT FRAC'S = 0
                                           ;11515          SC_K[ZERO],           :   RESULT SET TO 0
                                           ;11516          RC[T1]_K[ZERO],D_0,   :
                                           ;11517          SET.CC(INST),         :   SET COND CODES
U 0441, 0F18,D738,1980,F988,00F4,644C      ;11518          STATE0?               :   SEE IF WE WERE CALLED FROM POLYD
                                           ;11519
                                           ;11520  =***0
                                           ;11521          ;0--------------------;   NOT CALLED FROM POLYD
U 044C, 0000,003E,0180,F800,0000,0010      ;11522          RETURN10              :   WHAT A WASTE
                                           ;11523
                                           ;11524          ;1--------------------;   CALLED FROM POLYD
U 044D, 0000,003E,0180,F800,0000,0012      ;11525          RETURN12              :   TREAT AS A NON-UNDERFLOW
                                           ;11526  =;END
```

```
                                      ;11527  ;          ADDD/SUBD ALIGNMENT SUBROUTINE
                                      ;11528  ;          ENTER FROM ADDD.10 BRANCH TARGETS WITH <Q,D> = FRACTION TO BE ALIGNED,
                                      ;11529  ;          FE = SC = AMOUNT TO ALIGN IT, LC(EXP) = LARGER OF (SRC EXP, DST EXP)
                                      ;11530  ;          ENTRY IS AT ALNNEG IF <Q,D> SHOULD BE NEGATED FIRST, ELSE AT ALNPOS.
                                      ;11531
                                      ;11532  ;          THERE ARE SEVERAL RETURN POINTS:
                                      ;11533  ;          RETURN10          EXP DIFF < 32 - D HAS ALIGNED FRAC<L>, SC HAS LC(EXP)
                                      ;11534  ;          RETURN21          32<=EXP DIFF<64 - <Q,D> HAS ALIGNED FRAC, SC HAS LC(EXP)
                                      ;11535  ;          RETURN31          EXP DIFF > 63 - <Q,D>=<BIG OP<H>,SRC FRAC<H>>, SC=LC(EXP)
                                      ;11536  ; NOTE:  MAX EXP DIFF IS 63 (INSTEAD OF THE USUAL 57) BECAUSE POLYD USES ADDD
                                      ;11537
                                      ;11538            ;-----------------------;
U 080E, 081F,2000,0180,F800,0010,0816 ;11539  ALNNEG: D_0-D, CLK.UBCC           ;   NEGATE FRACTION - LOW ORDER FIRST
                                      ;11540
                                      ;11541            ;-----------------------;
U 0816, 001F,0300,01C0,F800,0000,04AC ;11542          Q_0-Q, C31?               ;   NEGATE HI-ORDER AND CHECK FOR BORROW
                                      ;11543
                                      ;11544  =0*       ;0----------------------;   BORROW
U 04AC, 0019,2000,05C0,F800,0000,04AE ;11545          Q_Q-K[.1]                 ;   OK, BORROW!
                                      ;11546
                                      ;11547            ;1----------------------;   NO BORROW
U 04AE, 0000,003C,8D80,F800,0094,881E ;11548  ALNPOS: SC_SC+K[.1F], CLK.UBCC    ;   SET EALU.N IF EXP DIFF => 32
                                      ;11549
                                      ;11550            ;-----------------------;
                                      ;11551          EALU_SC+K[.20], CLK.UBCC,;   SET EALU.N IF EXP DIFF > 63
U 081E, 0000,123C,7580,F800,0095,8546 ;11552             SC_FE, EALU?           ;   RESTORE ORIG EXP DIFF, TEST DIFF=>32
                                      ;11553
                                      ;11554  =0110
                                      ;11555            ;00---------------------;   EXP DIFF < 32, FRACTION POSITIVE
                                      ;11556          D_DAL.SC, SC_LC(EXP),     ;   D GETS LOW-ORDER ALIGNED FRAC
U 0546, 0D10,003A,0180,F800,0083,0010 ;11557                   RETURN[10]       ;
                                      ;11558
                                      ;11559            ;01---------------------;   EXP DIFF < 32, FRACTION NEGATIVE
                                      ;11560          D_DAL.SC, SC_LC(EXP),     ;   NEGATIVE SCHMEGATIVE - LET THE CALLER WORRY
U 0547, 0D10,003A,0180,F800,0083,0010 ;11561                   RETURN[10]
                                      ;11562
                                      ;11563            ;10---------------------;   EXP DIFF => 32, FRACTION POSITIVE
                                      ;11564          SC_SC+K[.20], D_Q, Q_0,   ;   SHIFT FRACT 32 PLACES RIGHT, REDUCE AMT BY 32
U 054E, 0C00,123C,75F8,F800,0084,8557 ;11565          EALU.N?, J/ALN.01         ;   TEST IF EXP DIFF > 63
                                      ;11566
                                      ;11567            ;11---------------------;   EXP DIFF => 32, FRACTION NEGATIVE
                                      ;11568          SC_SC+K[.20], D_Q,        ;   SHIFT FRAC 32 BITS RT, REDUCE SHFTCT BY 32
U 054F, 0C03,1228,75C0,F800,0084,8557 ;11569          ALU_-1, Q_ALU, EALU.N?    ;   SIGN-EXTEND FRACT, TEST IF EXP DIFF > 63
                                      ;11570  =;END
                                      ;11571
                                      ;11572  =0111     ;0----------------------;   32<= EXP DIFF <= 63
                                      ;11573  ALN.01: D_DAL.SC, SC_LC(EXP),     ;   <Q,D> HAVE ALIGNED SIGN-EXTENDED FRACT
U 0557, 0D10,003A,0180,F800,0083,0021 ;11574                   RETURN[21]       ;   SC HAS BIGGER EXP - RETURN
                                      ;11575
                                      ;11576            ;1----------------------;   63 < EXP DIFF
                                      ;11577          D_LC, SC_LC(EXP),         ;   HOPELESS - D GETS BIGGER OPERAND<H>,
U 055F, 0810,003A,C1F0,2C00,0083,0031 ;11578          Q_ID[T0], RETURN[31]      ;   SC GETS BIGGER EXP, Q GETS SRC FRAC<H>
```

```
                                        ;11579  .TOC    "       F & D floating point   : MULD"
                                        ;11580
                                        ;11581  ;MULD, *-R
                                        ;11582  ;ENTER HERE AT B.FORK WITH SRCO IN <RCO, D>, R IN LA, LB.
                                        ;11583  ;LEAVE HERE WITH SRCO IN <RCO, Q>, DSTO IN <RC1, D> TO GOTO SAME FLOW
                                        ;11584  ;AS *-*, *-*-*.
                                        ;11585
                                        ;11586  228:     ;-----------------------;
                                        ;11587           STATE_K[ZERO],          :  CLR POLYD FLAG
                                        ;11588           RC[T1]_LA,              :  RC1 GETS DSTO <H>
U 0228, 0000,003C,19E0,F988,1404,6325   ;11589           Q_D                     ,  Q GETS DSTO <L>
                                        ;11590
                                        ;11591  =0****   ;0----------------------;
                                        ;11592           D_R(PRN+1),             :  D GETS DSTO <L>
                                        ;11593           STATE_K[ZERO],          :  CLR FLAG FOR POLYD
U 0325, 0800,003D,1980,F860,1404,7003   ;11594           CALL,J/MULD.00          :  CALL MULD SUBROUTINE
                                        ;11595
                                        ;11596           ;1----------------------;  RETURN WITH RESULT IN <D, RC1>
                                        ;11597           R(PRN)_D,               :  STORE RESULT <H>
U 0335, 0001,003C,0180,F8D8,0000,0796   ;11598           J/ADDD.P                :  GOTO STORE RESULT <L>
                                        ;11599  =;END
                                        ;11600
                                        ;11601
                                        ;11602  ;DOUBLE FLOATING POINT ARETHMETIC MULD.
                                        ;11603  ;ENTER FROM DP WITH SRC OPD IN <RCO, Q>, DST OPD IN <RC1, D>.
                                        ;11604  ;ALWAYS YIELDS NORMALIZED AND ROUNDED RESULT.
                                        ;11605
                                        ;11606  38C:
                                        ;11607  MULD:    ;0----------------------;
                                        ;11608           STATE_K[ZERO],          :  CLR FLAG FOR POLYD
U 038C, 0000,003D,1980,F800,1404,7003   ;11609           CALL,J/MULD.00          :  CALL MULD SUBROUTINE
                                        ;11610
                                        ;11611  39C:     ;1----------------------;
U 039C, F000,003F,01F0,F847,0000,0300   ;11612           WRITE.DEST,J/WRD        :  WRITE RESULT
```

```
                                        ;11613  .TOC    "       F & D floating point   : DIVD"
                                        ;11614
                                        ;11615  ;DIVD, *-R
                                        ;11616  ;ENTER HERE AT B.FORK WITH SRCO IN <RCO, D>, R IN LA, LB.
                                        ;11617  ;LEAVE HERE WITH SRCO IN <RCO, Q>, DSTO IN <RC1, D> TO GOTO SAME FLOW
                                        ;11618  ;AS *-*, *-*-*.
                                        ;11619
                                        ;11620  229:    ;-----------  -----------;
                                        ;11621          RC[T1]_LA,               ;  RC1 GETS DSTO <H>
U 0229, 0000,003C,01E0,F988,0000,082A   ;11622          Q_D                      ;  Q GETS DSTO <L>
                                        ;11623
                                        ;11624          ;------------------------;
U 082A, 0800,003C,0180,F860,0000,0360   ;11625          D_R(PRN+1)               ;  D GETS DSTO <L>
                                        ;11626
                                        ;11627  =0****
                                        ;11628          ;0-----------------------;
                                        ;11629          RC[T6]_D,                ;  SAVE DSTO <L>
                                        ;11630          D_Q,                     ;  D GETS SRCO <L>
                                        ;11631          SC_K[.10],               ;  SC GETS 16. FOR SWAP WORD OF FRAC <L>
U 0360, 0C01,003D,6580,F9B0,0084,6608   ;11632          CALL,J/DIVD.S            ;  CALL DIVD SUBROUTINE
                                        ;11633
                                        ;11634          ;1-----------------------;  RETURN WITH RESULT IN <D, RC1>
                                        ;11635          ALU_D,N&Z_ALU.V&C_0,     ;  SET COND CODES N,Z
U 0370, 0001,5A3C,0180,F800,0050,00FC   ;11636          WORD,PSL.V?,J/ADDD.M     ;  GOTO STORE RESULT <H,L>
                                        ;11637  =;END
                                        ;11638
                                        ;11639
                                        ;11640  ;DOUBLE FLOATING POINT DIVD.
                                        ;11641  ;ENTER FROM DP WITH SRC OPD IN <RCO, Q>, DST OPD IN <RC1, D>.
                                        ;11642  ;QUOT = DST/SRC = <RC1, D>
                                        ;11643  ;ALWAYS YIELDS NORMALIZED AND ROUNDED RESULT.
                                        ;11644
                                        ;11645  38B:
                                        ;11646  DIVD:   ;------------------------;
                                        ;11647          RC[T6]_D,                ;  SAVE DSTO <L>
                                        ;11648          D_Q,                     ;  D GETS SRCO <L>
                                        ;11649          SC_K[.10],               ;  SC GETS 16. FOR SWAP WORD OF FRAC <L>
U 038B, 0C01,003D,6580,F9B0,0084,6608   ;11650          CALL, J/DIVD.S           ;  CALL DIVD SUBROUTINE
                                        ;11651
                                        ;11652  39B:    ;------------------------;
                                        ;11653          ALU_D,N&Z_ALU.V&C_0,     ;  SET COND CODES N,Z
U 039B, 0001,5A3C,0180,F800,0050,005C   ;11654          WORD,PSL.V?              ;  HAS TO RESET PSL<V>?
                                        ;11655
                                        ;11656  =110*   ;0-----------------------;  NO:
U 005C, F000,003F,01F0,F847,0000,0300   ;11657          WRITE.DEST,J/WRD         ;  WRITE RESULT
                                        ;11658
                                        ;11659          ;1-----------------------;  YES:
                                        ;11660          SET.V,                   ;  RESET PSL <V>
U 005E, F000,003F,01F0,F847,0020,0300   ;11661          WRITE.DEST,J/WRD         ;  WRITE RESULT
                                        ;11662  =;END
```

```
                                       ;DOUBLE FLOATING POINT DIVD ROUTINE.
                                       ;
                                       ;ALGORITHM:      NON-RESTORING DIVIDE IN TWO LOOPS, ONE FOR THE FIRST 32 BITS
                                       ;                OF QUOTIENT AND ONE FOR THE NEXT 26 BITS.
                                       ;
                                       ;INPUTS:         RC[T0] = SRC<H>
                                       ;                D = Q = SRC<L>
                                       ;                RC[T1] = DST<H>
                                       ;                RC[T6] = DST<H>
                                       ;                SC = 10 (HEX)
                                       ;
                                       ;OUTPUTS:        D = PACKED ROUNDED QUOTIENT<H>
                                       ;                RC[T1] = PACKED ROUNDED QUOTIENT<L>
                                       ;                CONDITION CODES AND ID[CES] SET UP FROM QUOTIENT
                                       ;
                                       ;TEMPORARIES:    R[R15], RC[T2], RC[T3], RC[T4], RC[T5],
                                       ;                ID[T0], ID[T1], ID[T2], ID[T3], ID[T4],
                                       ;                SS, SD, Q, FE, LA, LB, LC
                                       ;                (R[R1] IS USED INTERNALLY, BUT IT IS SAVED FIRST & RESTORED AFTER)
                                       ;
                                       ;RETURNS:        RETURNS @ 10
                                       ;
                                       =00
                                       DIVD.S: ;00---------------------;
                                               RC[T3]_D,                ;  SAVE SRC0 <L>
                                               D_DAL.SC,                ;  SWAP WORD OF SRC0 <L>
                                               SC_K[.FFF9],             ;  SETUP SHIFT AMOUNT -7
U 0608, 0D01,003D,BD80,F998,0084,66CA          CALL,J/UNPACK            ;  CALL UNPACK DOUBLE FLOATING PT OPERANDS ROUTINE

                                               ;01---------------------;  RETURN1, SRC = 0, DST MAY BE 0
                                               D_K[.8000],              ;  DIVISOR IS 0, NO DIV
                                               N&Z_ALU.V&C_0,WORD,      ;  COND CODES SET BY D'END
U 0609, 0818,4038,4580,F800,0050,0470          J/DIVD.20                ;

                                               ;10---------------------;  RETURN2, SRC.NE.0, DST = 0
                                               RC[T1]_K[ZERO], D_0,     ;  RESULT QUOTIENT IS 0
                                               N&Z_ALU.V&C_0,           ;  COND CODES SET BY D'END
U 060A, 0F18,003A,1980,F988,0050,0010          RETURN10                 ;  GOTO WRITE RESULT

                                               ;11---------------------;  RETURN3, SRC.NE.0, DST.NE.0
                                               ID[T4]_D,                ;  SAVE DST (D'END) FRAC <L>
                                               D_R[R1],                 ;  D GETS R1
U 060B, 0800,003C,D184,3E08,0081,082E          SD_SS, SC_FE             ;  SD, SC GET RESULT SIGN, EXP
                                       =;END

                                               ;-----------------------;
                                               SD_NOT.SD,               ;  FIX RESULT SIGN
                                               ID[T3]_D,                ;  SAVE R1
U 082E, 0001,203C,CD83,3EF8,0000,0832          R[R15]_Q                 ;  GET SRC (D'SOR) FRAC <L>

                                               ;-----------------------;
                                               Q_ID[T4],                ;  Q GETS DST (D'END) FRAC <L>
                                               LAB_R[R15],              ;  LB GETS SRC (D'SOR) FRAC <L>
U 0832, 0003,003C,D1F0,2E78,1408,6836          STATE_0(A)               ;  CLR POLYD FLAG
```

```
                                         ;11717               ;-----------------------;
                                         ;11718               Q_ID[T0],               ;  Q GETS D'SOR FRAC <H>
U 0836, 0001,203C,C1F0,2E88,0000,083A    ;11719               R[R1]_Q                 ;  R1 GETS D'END FRAC <L>
                                         ;11720
                                         ;11721               ;-----------------------;
                                         ;11722               RC[T4]_Q,               ;  RC4 GETS D'SOR <H>
U 083A, 0001,203C,C5F0,2DA0,0000,083E    ;11723               Q_ID[T1]                ;  Q GETS DST (D'END) FRAC <H>
                                         ;11724
                                         ;11725               ;-----------------------;
U 083E, 0000,003C,0180,F888,0000,084E    ;11726               LA_RA[R1]               ;  LATCH D'END FRAC <L>
                                         ;11727
                                         ;11728               ;-----------------------;
                                         ;11729               SC_SC+K[.80],           ;  ADD EXP BIAS 128. TO RESULT EXP
                                         ;11730               D_Q, Q_0,               ;  D GETS D'END FRAC <H>, CLR QUOT BITS
                                         ;11731               LC_RC[T4]&R1_(LA-LB).LEFT,;LC GETS D'SOR<H>, R1 GETS IMMED D'END FRAC<L>
                                         ;11732               SI/ZERO,                ;  SHIFT IN ZEROS
U 084E, 0C2C,0000,41F8,FBA0,00F4,8852    ;11733               SET.CC(LONG)            ;  SET PSL<N,C> BITS
                                         ;11734
                                         ;11735               ;-----------------------;
                                         ;11736               ALU_D[INST.DEP]LC,      ;  SUBT HIGH FRACS
                                         ;11737               D_ALU.LEFT.SI/DIVD,     ;
                                         ;11738               Q_Q.LEFT,               ;  SHIFT IN QUOT BIT
                                         ;11739               LA_RA[R1],              ;  LATCH D'END FRAC <L>
                                         ;11740               CLR.UBCC,               ;  FLAG FOR NEXT OPERATION, + OR -
U 0852, 0851,000C,5028,F888,0114,6861    ;11741               FE_K[.1E]               ;  SET LOOP CT FOR 31 LOOPS
                                         ;11742
                                         ;11743               ;-----------------------;
                                         ;11744               SC_FE,RC[T2]_K[SC],     ;  SC GETS 30., SAVE RESULT EXP IN RC2
U 0861, 0018,0338,1D80,F990,0081,04B0    ;11745               C31?                    ;  + OR - ?
                                         ;11746
                                         ;11747  =0*          ;0----------------------;
                                         ;11748               LC_RC[T4]&R1_(LA+LB).LEFT,;  R1 GETS IMMEDIATE D'END FRAC <L>
                                         ;11749               SI/ZERO,                ;  SHIFT IN ZEROS
                                         ;11750               SET.CC(LONG),           ;  SET PSL<N,C> BITS
                                         ;11751               SC_SC+1,                ;  INC COUNTER BY 1 SINCE 1ST QUOT BIT IS 0
U 04B0, 002C,0014,0180,FBA0,00F0,C89A    ;11752               J/DIVD.08               ;  GOTO ADD HIGH FRACS
                                         ;11753
                                         ;11754               ;1----------------------;
                                         ;11755               STATE_K[.80],           ;  SET FLAG TO INC EXP BY 1
                                         ;11756               LC_RC[T4]&R1_(LA-LB).LEFT,;  R1 GETS IMMEDIATE D'END FRAC <L>
                                         ;11757               SI/ZERO,                ;  SHIFT IN ZEROS
                                         ;11758               SET.CC(LONG),           ;  SET PSL<N,C> BITS
U 04B2, 002C,0000,4180,FBA0,1474,6865    ;11759               J/DIVD.06               ;  GOTO SUBT HIGH FRACS
                                         ;11760  =;END
                                         ;11761  =*0*
                                         ;11762  DIVD.04:;0----------------------;  FIRST 32 QUOT BITS DONE
                                         ;11763               SC_K[.1A],              ;  SHIFT AMOUNT FOR LOWER QUOTIENT BITS
                                         ;11764               R[R15]_Q,               ;  SAVE HIGH QUOT FRAC BITS
                                         ;11765               Q_0,                    ;  CLR QUOT FOR ADDITIONAL QUOT BITS
U 04BC, 0001,233C,E5F8,FAF8,0084,64DC    ;11766               C31?, J/DIVD.10         ;  + OR - FOR NXT QUOT BIT?
                                         ;11767
                                         ;11768               ;1----------------------;
                                         ;11769               FE_K[.8],               ;  PRE-SET SHIFT AMOUNT
U 04BE, 0000,033C,0180,F800,0104,64CC    ;11770               C31?                    ;  NXT OPERATION, + OR - ?
                                         ;11771  =;END
```

```
                                          ;11772  =0*       ;0----------------------;
                                          ;11773            LC_RC[T4]&R1_(LA+LB).LEFT,;  R1 GETS IMMEDIATE D'END FRAC <L>
                                          ;11774            SI7ZERO,                ;  SHIFT IN ZEROS
                                          ;11775            SET.CC(LONG),           ;  SET PSL<N,C> BITS
                                          ;11776            SC_SC-K[.1],            ;  DECREMENT COUNTER BY 1
U 04CC, 002C,0014,0580,FBA0,00F4,A89A     ;11777            J/DIVD.08               ;  GOTO ADD HIGH FRACS
                                          ;11778
                                          ;11779            ;1----------------------;
                                          ;11780            LC_RC[T4]&R1_(LA-LB).LEFT,;  R1 GETS IMMEDIATE D'END FRAC <L>
                                          ;11781            SI7ZERO,                ;  SHIFT IN ZEROS
                                          ;11782            SET.CC(LONG),           ;  SET PSL<N,C> BITS
U 04CE, 002C,0000,0580,FBA0,00F4,A865     ;11783            SC_SC-K[.1]             ;  DECREMENT LOOP COUNT
                                          ;11784  =;END
                                          ;11785  DIVD.06:;----------------------;
                                          ;11786            ALU_D[INST.DEP]LC,      ;  SUBT HIGH FRACS
                                          ;11787            D_ALU.LEFT,SI/DIVD,     ;
                                          ;11788            Q_Q.LEFT,               ;  SHIFT IN QUOT BIT
                                          ;11789            LA_RA[R1],              ;  LATCH D'END FRAC <L>
                                          ;11790            CLK.UBCC,               ;  FLAG FOR NEXT OPERATION, + OR -
U 0865, 0831,140C,0028,F888,0010,04BC     ;11791            SC.GT.0?, J/DIVD.04     ;  END OF 1ST LOOP?
                                          ;11792
                                          ;11793  DIVD.08:;----------------------;
                                          ;11794            ALU_D+LC+PSL.C,         ;  ADD HIGH FRACS
                                          ;11795            D_ALU.LEFT,SI/DIVD,     ;
                                          ;11796            Q_Q.LEFT,               ;  SHIFT IN QUOT BIT
                                          ;11797            LA_RA[R1],              ;  LATCH D'END FRAC <L>
                                          ;11798            CLK.UBCC,               ;  FLAG FOR NEXT OPERATION, + OR -
U 089A, 0831,142C,0028,F888,0010,04BC     ;11799            SC.GT.0?, J/DIVD.04     ;  END OF 1ST LOOP?
                                          ;11800
                                          ;11801  =0*
                                          ;11802  DIVD.10:;----------------------;
                                          ;11803            LC_RC[T4]&R1_(LA+LB).LEFT,;  R1 GETS IMMEDIATE D'END FRAC <L>
                                          ;11804            SI7ZERO,                ;  SHIFT IN ZEROS
                                          ;11805            SET.CC(LONG),           ;  SET PSL<N,C> BITS
                                          ;11806            SC_SC-K[.1],            ;  DECREMENT COUNTER BY 1
U 04DC, 002C,0014,0580,FBA0,00F4,A8A2     ;11807            J/DIVD.12               ;  GOTO ADD LOW FRACS
                                          ;11808
                                          ;11809            ;1----------------------;
                                          ;11810            LC_RC[T4]&R1_(LA-LB).LEFT,;  R1 GETS IMMEDIATE D'END FRAC <L>
                                          ;11811            SI7ZERO,                ;  SHIFT IN ZEROS
                                          ;11812            SET.CC(LONG),           ;  SET PSL<N,C> BITS
U 04DE, 002C,0000,0580,FBA0,00F4,A89E     ;11813            SC_SC-K[.1]             ;  DECREMENT COUNTER BY 1
                                          ;11814  =;END
                                          ;11815
                                          ;11816            ;----------------------;
                                          ;11817            ALU_D[INST.DEP]LC,      ;  SUBT HIGH FRACS
                                          ;11818            D_ALU.LEFT,SI/DIVD,     ;
                                          ;11819            Q_Q.LEFT,               ;  SHIFT IN QUOT BIT
                                          ;11820            LA_RA[R1],              ;  LATCH D'END FRAC <L>
                                          ;11821            CLK.UBCC,               ;  FLAG FOR NEXT OPERATION, + OR -
U 089E, 0831,140C,0028,F888,0010,04F0     ;11822            SC.GT.0?, J/DIVD.14     ;  END OF 2ND LOOP (LOW QUOT BITS) ?
```

```
                                   ;11823  DIVD.12:;-----------------------;
                                   ;11824          ALU_D+LC+PSL.C,         :  ADD HIGH FRACS
                                   ;11825          D_ALU.LEFT,SI/DIVD,     :
                                   ;11826          Q_Q.LEFT,               :  SHIFT IN QUOT BIT
                                   ;11827          LA_RA[R1],              :  LATCH D'END FRAC <L>
                                   ;11828          CLK.UBCC,               :  FLAG FOR NEXT OPERATION, + OR -
U 08A2, 0831,142C,0028,F888,0010,04F0  ;11829          SC.GT.0?                :  END OF 2ND LOOP (LOW QUOT BITS)?
                                   ;11830  =*0*
                                   ;11831  DIVD.14:;0----------------------;  YES: END OF LOOPS
                                   ;11832          D_Q,Q_ID[T3],           :  D GETS LOW QUOT BITS, Q GETS BACK OLD R1
                                   ;11833          LAB_R[R15],             :  LATCH HIGH QUOT BITS
U 04F0, 0C00,003C,CDF0,2E78,0000,08A6  ;11834          J/DIVD.16               :  GOTO PACKING
                                   ;11835
                                   ;11836          ;1----------------------;  NO: NOT END YET
U 04F2, 0000,033C,0180,F800,0000,04DC  ;11837          C31?, J/DIVD.10         :  GOTO +/- FOR NXT QUOT BIT
                                   ;11838  =;END
                                   ;11839
                                   ;11840  DIVD.16:;-----------------------;
                                   ;11841          R[R1]_Q,                :  RESTORE R1
U 08A6, 0001,203C,D5F8,FA88,0084,68AA  ;11842          SC_K[.6],Q_0            :  SET TO ALIGN LOW QUOT BITS
                                   ;11843
                                   ;11844          ;-----------------------;
                                   ;11845          D_DAL.SC,               :  SHIFT LOW QUOT BITS TO LEFT
                                   ;11846          SC_RC[T2],              :  SC GETS RESULT EXP
U 08AA, 0D10,0038,7D80,F910,0186,68BA  ;11847          FE_K[.18]               :  FE GETS 24. FOR NXT SHIFT
                                   ;11848
                                   ;11849          ;-----------------------;
                                   ;11850          Q_D,D_LA,K[.80],        :  D GETS BACK HIGH QUOT BITS
U 08BA, 0800,163C,41E0,F800,0000,02D4  ;11851          STATE7-4?               :  INC EXP BY 1?
                                   ;11852
                                   ;11853  =0***   ;0----------------------;
                                   ;11854          Q_Q+K[.80],CLK.UBCC,    :  Q GETS LOW QUOT FRAC BITS
U 02D4, 0019,2014,41C0,F800,0010,03FC  ;11855          J/PACKD                 :  GOTO PACKING
                                   ;11856
                                   ;11857          ;1----------------------;
                                   ;11858          SC_SC+1,                :  EXP ADJUSTED BY 1
                                   ;11859          Q_Q+K[.80],CLK.UBCC,    :  Q GETS LOW QUOT FRAC BITS
U 02DC, 0019,2014,41C0,F800,0090,C3FC  ;11860          J/PACKD                 :  GOTO PACKING
                                   ;11861  =;END
                                   ;11862
                                   ;11863  =0
                                   ;11864  DIVD.20:;0----------------------;
                                   ;11865          D_K[.8000],             :  FP -0
U 0470, 0818,0039,4580,F800,0000,0E00  ;11866          CALL,J/DIVBY0           :  GOTO SET CES FL DIV BY 0
                                   ;11867
                                   ;11868          ;1----------------------;
U 0471, 0000,003E,0180,F800,0000,0010  ;11869          RETURN10                :
                                   ;11870
                                   ;11871  ; **********************************************
                                   ;11872  ; * Patch no. 071, PCS 0471 trapped to WCS 1190 *
                                   ;11873  ; **********************************************
                                   ;11874  =;END
```

```
                                          .TOC    ''        F & D floating point   : UNPACK ONE DOUBLE OPERAND''

                                          ;INPUTS:          Q = OPERAND<H>
                                          ;                 D = OPERAND<L>

                                          ;OUTPUTS:         SC = EXPONENT
                                          ;                 SS = SD = SIGN
                                          ;                 ID[T1] = RC[T5] = HIGH FRACTION WITH NORMALIZE BIT
                                          ;                 D = LOW FRACTION

                                          ;TEMPORARIES:     R[R15], LA, LB

                                          ;RETURNS:         RETURNS @ 1 IF OPERAND = 0
                                          ;                 RETURNS @ 2 IF OPERAND NON-ZERO

                                          UNPK:     ;---------------------;
                                                    R[R15]_Q,             ;    R15 GETS OPR<H>
U 08BE, 0001,203C,65E0,FAF8,0084,68DA               Q_D,SC_K[.10]         ;    READY TO SWAP OPR <L> WORD

                                          UNPK.1:   ;---------------------;
                                                    D_DAL.SC,             ;    LOW FRAC WORD-SWAPPED
                                                    SC_K[.19],            ;    FOR FRAC <H> ALIGN
                                                    Q_R[R15](FRAC),       ;    OPR FRAC <H>
                                                    SS_SS.XOR.ALU15&SD_ALU15,;  SET SS/SD
U 08DA, 0D00,003C,B9CD,FA78,0884,68DE               CHK.FLT.OPR           ;    CHECK FOR OPR = -0

                                                    ;---------------------;
                                                    D_DAL.SC,             ;    RIGHT JUSTIFIED FRAC <H>
U 08DE, 0D00,003C,5DE0,F800,0084,68FD               Q_D,SC_K[.7]          ;    SET FOR LEFT JUSTIFIED FRAC <H>

                                                    ;---------------------;
                                                    D_DAL.SC,             ;    D GETS UNPACKED FRAC <H>
                                                    EALU_R[R15](EXP),     ;    CLOCK IN IF 0 EXP
U 08FD, 0D00,003C,0180,FA78,0018,6911               CLK.UBCC              ;

                                                    ;---------------------;
                                                    ID[T1]_D,RC[T5]_D,    ;    STORE UNPACKED FRAC <H>
                                                    D_Q,Q_0,              ;    READY FOR FRAC <L>
U 0911, 0C01,123C,C5F8,3DA8,0000,03E3               EALU.Z?               ;    EXP 0?

                                          =*011     ;0--------------------;    NO: EXP .NE. 0
                                                    D_DAL.SC,             ;    D GETS UNPACKED FRAC <L>
                                                    SC_R[R15](EXP),       ;    SC GETS EXP
U 03E3, 0D00,003E,0180,FA78,0083,0002               RETURN2               ;

                                                    ;1--------------------;    YES: EXP = 0
                                                    RC[T5]_K[ZERO],D_0,   ;    RESULT 0
                                                    SC_K[ZERO],           ;
U 03E7, 0F18,003A,1980,F9A8,0084,6001               RETURN1               ;
                                          =;END
```

```
                                        ;11925  .TOC    "       F & D floating point   : CVTBF, CVTWF, CVTLF"
                                        ;11926
                                        ;11927  ;         CVTB/W/LF        (4C/4D/4E)       SRC.RX, DST.WF
                                        ;11928  ;ENTER HERE AT B.FORK, WITH D CONTAINS SRC
                                        ;11929
                                        ;11930  24F:      ;---------------------------;
                                        ;11931  CVTBF:    D_D.SXT[INST.DEP],Q_0,      ;   SIGN EXT SRC TO D
                                        ;11932            SET.CC(INST),               ;   SET COND CODES
                                        ;11933            CLR.SD&SS,                  ;   ASSUME POSITIVE RESULTS
                                        ;11934            SC_K[.A0],                  ;   SC SET TO BIAS 128 + SHF 32.
U 024F, 0802,C93C,25FF,F800,00F4,65BA   ;11935            IRT?                        ;   IS IT CVTLF?
                                        ;11936
                                        ;11937  =10       ;0--------------------------;   NO: IT IS CVTBF/CVTWF
                                        ;11938  CVTBF.0:FE_SC-SHF.VAL,               ;   ADJUST EXP FOR NORALIZATION
                                        ;11939            D_DAL.NORM,                 ;   D GETS NORMALIZED FRAC
U 05BA, 0E00,0D3C,0180,F800,010C,A610   ;11940            SIGNS?,J/CVTBF.1            ;   BEN ON D.NE.0, D[31]
                                        ;11941
                                        ;11942            ;1--------------------------;   YES: CVTLF
U 05BB, 0000,0D3C,0180,F800,0000,0650   ;11943            SIGNS?,J/CVTLF.2            ;   TEST SRC SIGN AND ZERONESS
                                        ;11944  =;END
                                        ;11945
                                        ;11946  =00       ;00-------------------------;   CONSTRAINT FOR SIGNS; ALSO C31 (SEE CVTBF.5+2)
U 0610, FC00,003F,01F0,F847,0000,0300   ;11947  CVTBF.1:D_Q, WRITE.DEST              ;   SRC=0 (OR NO OVFLO ON ROUND - SEE CVTBF.5+2)
                                        ;11948
                                        ;11949  =10       ;10-------------------------;
                                        ;11950            EALU_FE,D_PACK.FP,          ;   SRC IS POS: D GETS PACKED FP RESULT
U 0612, F808,003B,01F0,F847,0000,6300   ;11951            W..ITE.DEST                 ;   GOTO WRITE RESULT
                                        ;11952
                                        ;11953            ;11-------------------------;
                                        ;11954            D_0-D,                      ;   NEGATIVE SOURCE.  MAKE IT POSITIVE
                                        ;11955            SD_NOT.SD,                  ;   SET RESULT SIGN
U 0613, 081F,2000,0183,F800,0000,05BA   ;11956            J/CVTBF.0                   ;   NORMALIZE AND STORE THAT
                                        ;11957  =;END
                                        ;11958  ;HERE FOR CONVERTS OF LONG TO FLOATING, WHICH MUST ROUND IF
                                        ;11959  ; THE SIGNIFICANT BITS OF INTEGER DO NOT ALL FIT IN THE FRACTION.
                                        ;11960
                                        ;11961  =00      ;00---------------------;  D.EQL.0
U 0650, F000,003F,01F0,F847,0000,0300   ;11962  CVTLF.2:WRITE.DEST               ;  SRC ZERO
                                        ;11963
                                        ;11964  =10      ;10---------------------;  D.GTR.0
                                        ;11965  CVTLF.3:SC_SC-SHF.VAL,           ;  SRC POS: ADJUST EXP FOR NORMALIZATION
                                        ;11966          D_DAL.NORM,              ;  D GETS NORMALIZED FRAC
                                        ;11967          K[.80],                  ;  PRESET KMX FOR SK FOR ROUNDING
U 0652, 0E00,183C,4180,F800,008C,A567   ;11968          D.BYTES?,J/CVTLF.4       ;  BEN ON D.NE.0, GOTO ROUNDING
                                        ;11969
                                        ;11970          ;11---------------------;  D.LSS.0
                                        ;11971          D_0-D,                   ;  NEGATE SOURCE
                                        ;11972          SD_NOT.SD,               ;  SET SIGN FLAG
U 0653, 081F,2000,0183,F800,0000,0652   ;11973          J/CVTLF.3                ;  THEN NORMALIZE
                                        ;11974  =;END
                                        ;11975
                                        ;11976  =0111    ;0----------------------;  D<31:24> .EQL. 0 (ROUNDING UNNECESSARY)
                                        ;11977  CVTLF.4:EALU_SC,D_PACK.FP,        ;  PREPARE RESULT
U 0567, F808,003B,01F0,F847,0000,0300   ;11978          WRITE.DEST               ;  STORE IT
                                        ;11979
                                        ;11980          ;1----------------------;
                                        ;11981          D_D+K[.80],CLK.UBCC,     ;  ROUND THE FRAC
U 056F, 0819,0014,4180,F800,0110,C915   ;11982          FE_SC+1                  ;  INC EXP TO FE IN CASE CARRY FROM ROUNDING
                                        ;11983  =;END
                                        ;11984
                                        ;11985          ;-----------------------;
                                        ;11986          EALU_SC,Q_PACK.FP,       ;  PACK RESULT FP AS IF THERE IS NO CARRY
U 0915, 0008,0338,01C0,F800,0000,0610   ;11987          C31?, J/CVTBF.1          ;  BEN ON ALU<C>
```

```
                                        ;11988  .TOC    ''      F & D floating point   : CVTBD, CVTWD, CVTLD''
                                        ;11989
                                        ;11990  ;       CVTB/W/LD       (6C/6D/6E)      SRC.RX, DST.WD
                                        ;11991  ;ENTER HERE WITH SRC IN D
                                        ;11992
                                        ;11993  2CC:    ;---------------------------;
                                        ;11994  CVTID:  D_D.SXT[INST.DEP],Q_0,    ;  SIGN EXT SRC TO D
                                        ;11995          SET.CC(INST),             ;  SET COND CODES
                                        ;11996          CLR.SD&SS,                ;  INIT RESULT SIGN TO POSITIVE
U 02CC, 0802,C03C,25FF,F800,00F4,6919   ;11997          SC_K[.A0]                 ;  SC SET TO BIAS 128 + SHF 32
                                        ;11998
                                        ;11999          ;---------------------------;
                                        ;12000          RC[T1]_0,                 ;  ASSUME ONLY 24 BITS OF FRACTION
                                        ;12001          FE_K[.8],                 ;  GET CONSTANT FOR RECOVERING OTHERS
U 0919, 0003,0D3C,0180,F988,0104,6668   ;12002          SIGNS?                    ;  TEST FOR SRC 0, +, OR -
                                        ;12003
                                        ;12004  =00     ;---------------------------;  D .EQL. 0
U 0668, F000,003F,01F0,F847,0000,0300   ;12005          WRITE.DEST,J/WRD          ;  EASY CASE
                                        ;12006
                                        ;12007  =10     ;---------------------------;  D .GTR. 0
                                        ;12008  CVTID.1:D_DAL.NORM,               ;  NORMALIZE FRACTION
                                        ;12009          FE_SC-SHF.VAL,            ;  CALCULATE EXPONENT
                                        ;12010          SC_FE,                    ;  GET 8 INTO SC
U 066A, 0E00,183C,0180,F800,018D,A577   ;12011          D.BYTES?,J/CVTID.2        ;  TEST BYTE 3 FOR MORE THAN 24 BITS OF FRAC
                                        ;12012
                                        ;12013          ;---------------------------;  D .LSS. 0
                                        ;12014          D_0-D,                    ;  NEGATE FRACTION
                                        ;12015          SD_NOT.SD,                ;  SET DESTINATION SIGN FLAG
U 066B, 081F,2000,0183,F800,0000,066A   ;12016          J/CVTID.1                 ;  GO NORMALIZE AND STORE
                                        ;12017
                                        ;12018  =;END OF CONSTRAINT ON BEN/SIGNS
                                        ;12019
                                        ;12020  =0111   ;---------------------------;  D<31:24> .EQL. 0 (RESULT FITS IN 1 LONGWORD)
                                        ;12021  CVTID.2:EALU_FE,D_PACK.FP,        ;  PACK FRAC AND EXP INTO D
U 0577, F808,003B,01F0,F847,0000,6300   ;12022          WRITE.DEST                ;  STORE IT AS DESTINATION
                                        ;12023
                                        ;12024          ;---------------------------;  D<31:24> .NEQ. 0 BEFORE NORMALIZATION
                                        ;12025          Q_D,                      ;  SAVE HIGH ORDER A MOMENT
U 057F, 0D00,003C,01E0,F800,0000,0920   ;12026          D_DAL.SC                  ;  SHIFT LOW ORDER LEFT 8 PLACES
                                        ;12027
                                        ;12028          ;---------------------------;
                                        ;12029          RC[T1]_D.AND.K[.FF00],    ;  STORE LOW ORDER FRACTION BITS
                                        ;12030          D_Q,                      ;  GET BACK HIGH ORDER
U 0920, 0C19,0034,4D80,F988,0000,0577   ;12031          J/CVTID.2
```

```
                                    .TOC    ''      F & D floating point  : CVTFD, CVTDF''

                                    ;CVTFD  (56)    SRC.RF, DST.WD
                                    ;ENTER WITH SRC IN D.

                                    247:        ;-----------------------;
                                    CVTFD:      SC_D(EXP), ALU_D,       ;  GET EXP
                                                SET.CC(INST),           ;  SET COND CODES
U 0247, 0001,C03C,0180,F800,08F3,0922           CHK.FLT.OPR             ;  CHK -0

                                                ;-----------------------;
                                                RC[T1]_K[ZERO],         ;  RESULT <L> IS ALWAYS 0
U 0922, 0018,1438,1980,F988,0000,03F9           SC?                     ;  CHK FOR +0

                                    =*01        ;0----------------------;
U 03F9, FF00,003F,01F0,F847,0000,0300 CVTFD.1:  D_0, WRITE.DEST         ; COND CODES ALREADY SHOW 0

                                                ;1----------------------;
U 03FB, F000,003F,01F0,F847,0000,0300           WRITE.DEST              ;SOURCE NON-ZERO - WRITE DEST
                                    =;END
```

```
                                        ;CVTDF  (76)     SRC.RD, DST.WF
                                        ;ENTER WITH SRC IN <RC 0, D>,

                                        2CD:     ;---------------------;
                                        CVTDF:   LC_RC[T0],            ;  LATCH SRC <H>
                                                 ALU_D.AND.K[.8000],   ;  CHK FOR NEED OF ROUNDING
U 02CD, 0019,0034,4580,F900,0010,0926            CLK.UBCC              ;

                                                 ;---------------------;
                                                 D_LC(FRAC),           ;  GET FRAC
                                                 SC_LC(EXP),           ;  GET EXP
                                                 CHK.FLT.OPR,          ;  CHK IF -0
                                                 SS_SS.XOR.ALU15&SD_ALU15,;  SD GETS SIGN
                                                 K[.80],               ;  PRESET SLOW CONSTANT FOR POSSIBLE ROUNDING
U 0926, 0910,0138,4185,F800,0883,04F4            Z?                    ;  HAVE TO DO ROUNDING?

                                        =0       ;0--------------------;  YES: ROUNDING
                                                 FE_SC,                ;  FE GETS OLD EXP
                                                 D_D+K[.80],           ;  ADD 1 TO FRAC
                                                 CLK.UBCC,             ;  HAVE TO INCREMENT EXP BY 1?
U 04F4, 0819,1414,4180,F800,0110,0449            SC?, J/CVTDF.1        ;  CHECK FOR ZERO SOURCE

                                        ; ***********************************************
                                        ; * Patch no. 012, FCS 04F4 trapped to WCS 114C *
                                        ; ***********************************************

                                                 ;1--------------------;  NO: RESULT = SRC <H>
                                                 D_LC.SET.CC(INST),    ;  SET COND CODES
U 04F5, 0810,D438,0180,F800,0070,03F9            SC?,J/CVTFD.1         ;  CHK FOR ZERO
                                        =;END

                                        =*01     ;0--------------------; SOURCE WAS ZERO DESPITE RANDOM LOW BITS
                                        CVTDF.1: D_0, ALU_K[ZERO],     ;SOURCE=0 --> DEST = 0
                                                 SET.CC(INST),         ;SET CC'S TO REFLECT A ZERO
U 0449, FF18,C03B,19F0,F847,0070,0300            WRITE.DEST            ;AND GO STORE IT

                                                 ;1--------------------;
                                                 SC_SC+1,              ;  POSSIBLE INCREMENT EXP
U 044B, 0000,183C,0180,F800,0080,C434            ALU.N?                ;  HAVE TO INCREMENT EXP?
                                        =;END

                                        =0***
                                        CVTDF.2:
                                                 ;0--------------------;  NO:
                                                 SC_SC-K[.1],          ;  GET BACK OLD EXP
U 0434, 0500,003C,0580,F800,0084,A43C            D_D.LEFT.SI/ZERO      ;  LEFT JUSTIFIED FRAC

                                                 ;1--------------------;  YES:
                                                 EALU_SC,              ;  EXP WAS INCREMENTED
                                                 D_PACK.FP,            ;  PACK RESULT
                                                 SET.CC(INST),         ;  SET COND CODES
U 043C, 0808,D438,0180,F800,0070,0601            SC?,J/EXPCK           ;  CHK FOR OVERFLOW
                                        =;END
```

```
                                      ;12105  .TOC    ''      F & D floating point  : CVTFB, CVTFW, CVTFL, CVTRFL''
                                      ;12106
                                      ;12107  ;       CVTFB/W/L        (48/49/4A)         SRC.RF, DST.WX
                                      ;12108  ;ENTER AT B.FORK WITH SRC IN D.
                                      ;12109
                                      ;12110  245:
                                      ;12111  CVTFB:  ;-----------------------;
                                      ;12112          D_D(FRAC),              ;  UNPACK FLOATING
                                      ;12113          SC_D(EXP),SS_ALU15,     ;  GET EXP, SIGN
                                      ;12114          CHK.FLT.OPR,            ;  CHECK FOR -0
U 0245, 0901,003D,0181,F800,0883,0944 ;12115          CALL,J/CVTFI.0          ;
                                      ;12116
                                      ;12117  255:    ;-----------------------;
                                      ;12118  CVTFX.W:ALU_D, N_AMX.Z_TST,     ;  SET COND CODES
U 0255, F001,C03F,01F0,F847,0030,0300 ;12119          DT/INST.DEP, WRITE.DEST ;  WRITE RESULT
                                      ;12120
                                      ;12121  ; ***********************************************
                                      ;12122  ; * Patch no. 015, PCS 0255 trapped to WCS 1152 *
                                      ;12123  ; ***********************************************
                                      ;12124
                                      ;12125
                                      ;12126  ;       CVTRFL           (4B)               SRC.RF, DST.WX
                                      ;12127  ;ENTER AT B.FORK WITH SRC IN D.
                                      ;12128
                                      ;12129  2C1:
                                      ;12130  CVTRFL: ;-----------------------;
                                      ;12131          D_D(FRAC),              ;  SAVE INPUT, UNPACK FLOATING
                                      ;12132          SC_D(EXP),SS_ALU15,     ;  GET EXP, SIGN
                                      ;12133          CHK.FLT.OPR,            ;  CHECK FOR -0
U 02C1, 0901,003D,0181,F800,0883,0944 ;12134          CALL,J/CVTFI.0          ;
                                      ;12135
                                      ;12136  2D1:    ;-----------------------;  RETURN FOR CVTRFL
                                      ;12137          ALU_0(A), Q_ALU.LEFT,   ;  MOVE ROUND BIT FROM Q31 TO Q00
U 02D1, 0023,123C,02C0,F800,0000,016E ;12138          SI/DIV, SS?             ;  WITH Q<31-1>=0 ; TEST SIGN
                                      ;12139
                                      ;12140  =1110   ;0----------------------;
U 016E, 081D,0014,0180,F800,0000,0255 ;12141          D_D+Q, J/CVTFX.W        ;  ROUND POSITIVE NUMBERS UP
                                      ;12142
                                      ;12143          ;1----------------------;
U 016F, 081D,0000,0180,F800,0000,0255 ;12144          D_D-Q, J/CVTFX.W        ;  ROUND NEGATIVE NUMBERS DOWN
                                      ;12145  =;END
```

```
                                        ;12146  .TOC    "       F & D floating point   : CVTDB, CVTDW, CVTDL, CVTRDL"
                                        ;12147
                                        ;12148  ;CVTDB/W/L      (68/69/6A)      SRC.RD, DST.WX
                                        ;12149  ;ENTER AT B.FORK WITH SRC IN <RC 0,  D>.
                                        ;12150  243:
                                        ;12151  CVTDB:  ;-----------------------;
                                        ;12152          Q_RC[T0],               ;  GET SRC <H>
U 0243, 0010,0039,01C0,F900,0000,0680   ;12153          CALL,J/CVTDI            ;  CALL CVT DOUBLE TO INTEGER SUBRT
                                        ;12154
                                        ;12155  253:    ;-----------------------;
                                        ;12156          ALU_D, N_AMX.Z_TST,     ;  SET CONDITION CODES
U 0253, F001,C03F,01F0,F847,0030,0300   ;12157          DT/INST.DEP, WRITE.DEST ;  WRITE RESULT
                                        ;12158
                                        ;12159  ; ************************************************
                                        ;12160  ; * Patch no. 015, PCS 0253 trapped to WCS 1152 *
                                        ;12161  ; ************************************************
```

```
                                            ;CVTRDL          (6B)              SRC.RD, DST.WX
                                            ;ENTER AT B.FORK WITH SRC IN <RC 0,  D>.
                                            2C5:
                                            CVTRDL: ;-----------------------;
                                                    Q_RC[T0],               ;  GET SRC <H>
U 02C5, 0010,0039,01C0,F900,0000,0680               CALL,J/CVTDI            ;  CALL CVT DOUBLE TO INTEGER SUBRT

                                             2D5:   ;-----------------------;
                                                    ALU_O(A), Q_ALU.LEFT,   ;ISOLATE ROUND BIT IN Q00
U 02D5, 0023,123C,02C0,F800,0000,041E               SI/DIV, SS?             ;  AL BY ITSELF ; TEST SIGN

                                            =1110
                                            CVTRDL.0:
                                                    ;0----------------------;
                                                    D_D+Q, Q_Q.RIGHT,       ;  ROUND POSITIVE NUMBERS UP, SET
                                                    SI/ZERO, CLK.UBCC,      ;  Q = 0
U 041E, 081D,0014,01B0,F800,0010,092E               J/CVTRDL.1

                                                    ;1----------------------;
                                                    D_D-Q, Q_Q.RIGHT,       ;  ROUND NEGATIVE NUMBERS DOWN, SET
U 041F, 081D,0000,03B0,F800,0010,092E               SI/MUL-, CLK.UBCC       ;  Q = 80000000
                                            =;END
                                            CVTRDL.1:
                                                    ;-----------------------;
U 092E, 001D,0120,01C0,F800,0000,0472               Q_D.XOR.Q, Z?           ;  COMPARE INPUT VS RESULT SIGN

                                            =010    ;0----------------------;  CONSTRAINT FOR Z AND Q31
U 0472, 0000,0D3C,0180,F800,0000,0473               Q31?                    ;  CHECK IF INPUT AND RESULT SIGNS ARE EQUAL

                                            =011    ;01---------------------;
                                            CVTRDL.2:
                                                    ALU_D, N_AMX.Z_TST,     ;  SET CONDITION CODES FROM RESULT
U 0473, F001,003F,01F0,F847,0030,0300               DT/LONG, WRITE.DEST     ;  AND GO WRITE IT

                                            ; ************************************************
                                            ; * Patch no. 015, PCS 0473 trapped to WCS 1152 *
                                            ; ************************************************

                                            =111    ;11---------------------;
U 0477, 0000,003C,0180,F800,0020,0473               SET.V, J/CVTRDL.2       ;  SIGNS UNLIKE AND RESULT.NE.0 MEANS OVFLO
                                            =;END
```

```
;12203  .TOC    ''      F & D floating point  : CONVERT FLOATING TO INTEGER''
;12204
;12205  :       GENERAL UTILITY ROUTINES FOR FLOATING-TO-INTEGER CONVERSION.
;12206  :       USED BY CVTFX, CVTDX, CVTRFL, CVTRDL, EMODF, EMODD INSTRUCTIONS.
;12207  :
;12208  :       ENTRY POINTS:
;12209  :
;12210  :       CVTFX, CVTRFL ENTER AT CVTFI.0 WITH D=UNPACKED MANTISSA,
;12211  :       SC = EXPONENT, SS = SIGN OF THE NUMBER TO CONVERT. RESERVED OPERAND
;12212  :       CHECK MUST HAVE ALREADY BEEN PERFORMED.
;12213  :       EMODF ENTERS AT CVTFI.1 WITH THE SAME PARAMETERS, EXCEPT THAT
;12214  :       THE OPERATIONS PERFORMED IN CVTFI.0 HAVE ALREADY BEEN DONE;
;12215  :       THIS IS TO AVOID LOSS OF PRECISION IN THE 32-BIT PRODUCT.
;12216  :
;12217  :       CVTDX, CVTRDL ENTER AT CVTDI WITH PACKED DOUBLE PRECISION NUMBER
;12218  :       IN <RC 0 -- Q , D>;
;12219  :       EMODD ENTERS AT CVTFI.1 WITH LOW FRACTION IN Q
;12220  :       EXIT CONDITIONS OF THIS ROUTINE:
;12221  :
;12222  :       EXIT IS VIA RETURN10 WITH D = INTEGER PART OF NUMBER (SIGNED),
;12223  :       Q31 = ROUND BIT (UNSIGNED), <RC[T1],RC[T2]> = MANTISSA OF
;12224  :       ORIGINAL F.P NUMBER IN NORMALIZED FORM (IF INTEGER OVERFLOWS,
;12225  :       THIS MANTISSA MAY BE SHIFTED LEFT 32 BITS OR MAY BE 0),
;12226  :       SC = AMOUNT TO SHIFT <RC[T1],RC[T2]> LEFT BY TO YIELD
;12227  :       THE FRACTIONAL PART OF THE NUMBER, SS = SIGN OF INPUT NUMBER.
;12228  :       THE CONDITION CODES N,Z, AND C ARE 0, 1 AND 0 RESPECTIVELY
;12229  :       AND V INDICATES WHETHER AN OVERFLOW HAS BEEN DETECTED
;12230  :       (OVERFLOW DETECTION IS DONE IN INSTRUCTION'S DATA TYPE)
```

```
                                          CVTFI.0:
                                                  ;-----------------------------;
                                                  RC[T1]_0, N&Z_ALU.V&C_0, Q_0,   ; CLEAR MANTISSA HOLDER, SET Z
                                                  D_D.LEFT,                       ; CHANGE FRAC FROM UNPACKED TO NRMLIZED
U 0944, 0503,003C,41FC,F988,01D4,A946             SD_SS, SC_SC-K[.80], FE_EALU    ; STRIP BIAS FROM EXP
                                          CVTFI.1:
                                                  ;-----------------------------; EMODF ENTERS HERE
                                                  RC[T2]_Q, Q_0,                  ; RC2 <- MANTISSA<L>, CLR Q FOR RT SHIFT
U 0946, 0001,343C,75F8,F990,0084,A4D4             SC_SC-K[.20], SC?               ; TURN EXP INTO SHIFT CT AND TEST RANGE

                                          =100                                    ; BRANCH ON SC (RANGE OF /NUM/)
                                          CVTFI.2:                                ;
                                                  ;00---------------------------; [CVTDI STUFF ENTERS HERE]
                                                  Q_D, RC[T1]_D, D_0,             ; .5<=/NUM/<1.0 - SAVE HI MANTISSA,
U 04D4, 0F01,003E,01E0,F988,0081,0010             SC_FE, RETURN10                 ; INT = 0, ROUND BIT = 1, EXIT.

                                                  ;01---------------------------;
                                                  RC[T1]_D, D_0, Q_0,             ; /NUM/<.5 - SAVE HI MANTISSA,
U 04D5, 0F01,003E,01F8,F988,0081,0010             SC_FE, RETURN10                 ; INT=0, ROUND=0, EXIT WITH SC<0.

                                                  ;10---------------------------;
                                                  RC[T1]_D, Q_D, D_DAL.SC,        ; 1<=/NUM/<2**31 - SAVE HI MANTISSA,
U 04D6, 0D01,123C,01E0,F988,0000,046E             SS?, J/CVTFI.4                  ; D=INT, SET UP TO GET ROUND BIT,
                                                                                  ; SEE IF WE SHOULD NEGATE INT.

                                                  ;11---------------------------;
U 04D7, 0010,0038,01C0,F910,0000,0952             Q_RC[T2]                        ;/NUM/=>2**31 - RESTORE Q FOR LEFT SHFT

                                                  ;-----------------------------;
                                                  EALU_SC, FE_EALU,               ; FORM MAGIC # TO
                                                  Q_D.OXT[BYTE].OR.PACK.FP,       ; CHECK SPECIAL CASE (NUM=-2**31),
U 0952, 0DCB,9430,01C0,F800,0100,04A6             D_DAL.SC, SC?                   ; SHIFT D AS IF SC<32 & TEST IF TRUE

                                          =110                                    ;BRANCH ON SC (/NUM/ => 2**63)
                                                  ;0----------------------------;
                                                  ALU_Q.XOR.K[.8000], CLK.UBCC,   ; 2**31<=/NUM/<2**63 - SEE IF
                                                  LC_RC[T2], Q_0,                 ; NUM=-2**31 (ONLY NON-OVERFLOW CASE)
U 04A6, 0019,2020,45F8,F910,0110,0956             FE_SC, J/CVTFI.3                ; DECREMENT SAVED SC BY 32

                                                  ;1----------------------------;
                                                  RC[T1]_LC, D_LC, Q_0,           ; /NUM/>=2**63 - SHIFT MANTISSA LEFT 32
U 04A7, 0810,0038,01F8,F988,0020,0946             SET.V, J/CVTFI.1                ; LOOP UNTIL NUMBER IS SHIFTABLE

                                          CVTFI.3:
                                                  ;-----------------------------; CONTINUATION OF /NUM/=>2**31 CASE
                                                  RC[T1]_LC, Q_LC, SC_SC-K[.20],  ; SHIFT MANTISSA LEFT 32 & ADJUST EXP
U 0956, 0010,0138,75C0,F988,0084,A504             Z?                              ; CHECK FOR NUM=-2**31
                                          =0
                                                  ;0----------------------------;BRANCH ON Z (NUM = 2**31)
U 0504, 0003,123C,0180,F990,0020,046E             RC[T2]_0, SET.V, SS?, J/CVTFI.4 ; NO - OVERFLOW - COMPLETE THE LEFT
                                                                                  ; SHIFT AND CHECK SIGN FOR NEGATION

                                                  ;1----------------------------;
U 0505, 0003,123C,0180,F990,0000,046E             RC[T2]_0, SS?, J/CVTFI.4        ; YES - NO OVERFLOW - COMPLETE THE LEFT
                                                                                  ; SHIFT AND TEST FOR NEGATION
```

```
                                        ;12286  ;       FLOW CONVERGES HERE WITH D = LOW 32 BITS OF INTEGER,
                                        ;12287  ;       Q = MANTISSA WORD CONTAINING ROUND BIT, SC=NUMBER OF PLACES
                                        ;12288  ;       TO SHIFT (RIGHT) TO PUT ROUND BIT IN D31, V SET IF OVERFLOW,
                                        ;12289  ;       <RC[T1],RC[T2]> CONTAINING MANTISSA SUCH THAT Q=RC[T1].
                                        ;12290  ;       WHAT WE HAVE TO DO IS GET THE ROUND BIT AND CHECK FOR
                                        ;12291  ;       DATA-TYPE OVERFLOW OF THE INTEGER.
                                        ;12292
                                        ;12293          ;--------------------------------;
                                        ;12294  =1110   ;BRANCH ON SS (ORIGINAL NUMBER NEGATIVE)
                                        ;12295          ;0-------------------------------;
                                        ;12296  CVTFI.4:
                                        ;12297          RC[T3]_D.SXT[INST.DEP],          ; CONVERT LONGWORD TO TARGET D.T.
U 046E, 0D02,C03C,01E0,F998,0000,095E   ;12298              Q_D, D_DAL.SC, J/CVTFI.5     ; SAVE INT AND GET ROUND BIT
                                        ;12299          ;1-------------------------------;
U 046F, 0D1F,2000,01C0,F800,0000,095A   ;12300          Q_0-D, D_DAL.SC                  ; NEGATE & SAVE INT, GET ROUND BIT
                                        ;12301          ;--------------------------------;
U 095A, 0002,E03C,0180,F998,00C0,095E   ;12302          RC[T3]_Q.SXT[INST.DEP]           ; CONVERT LONGWORD TO TARGET D.T.
                                        ;12303          ;--------------------------------;
                                        ;12304  CVTFI.5:
                                        ;12305          LC_RC[T3], ALU_Q.XOR.LC,         ; CHECK FOR D.T. OVERFLOW
U 095E, 0C11,2020,01E0,F918,0091,096C   ;12306              CLK.UBCC, D_Q, Q_D, SC_FE    ; D<-INT, Q<-ROUND, SC<-# BITS
                                        ;12307          ;--------------------------------;
U 096C, 0000,013C,0180,F800,0000,0544   ;12308          Z?                               ; CHECK FOR OVERFLOW (WHADDA WASTE)
                                        ;12309          ;--------------------------------;
                                        ;12310  =0      ;BRANCH ON Z (NO OVERFLOW)
                                        ;12311          ;0-------------------------------;
U 0544, 0000,003E,0180,F800,0020,0010   ;12312          SET.V, RETURN10                  ; OVERFLOW - RETURN WITH V=1
                                        ;12313          ;1-------------------------------;
U 0545, 0000,003E,0180,F800,0000,0010   ;12314          RETURN10                         ; NO OVERFLOW (HOWEVER V STILL MAY BE 1)
                                        ;12315          ;--------------------------------;
```

```
                                    ;       DOUBLE PRECISION ENTRY POINT

                                            ;-------------------------------;
                                    =00     ;CALL CONSTRAINT FOR UNPK
                                            ;00-----------------------------;
                                    CVTDI:  RC[T1]_0, N&Z_ALU.V&C_0,        ; INIT HI FRAC AND COND CODES,
                                            SS_0&SD_0,                      ; SET UP FOR UNPACK ROUTINE
U 0680, 0003,003D,0187,F988,0050,088E       CALL, J7UNPK                    ; GO UNPACK DOUBLE PREC NUMBER

                                            ;01-----------------------------;
                                            D_0, Q_0, RC[T2]_0, SC_K[ZERO], ; NUMBER = 0 - ZERO WORLD
U 0681, 0F03,003E,19F8,F990,0084,6010               RETURN10                ; AND EXIT

                                            ;10-----------------------------; NUM .NE. 0
                                            D_RC[T5], Q_D, SD_SS,           ;GET UNPACKED MANTISSA IN <D,Q>
U 0682, 0810,0038,41E4,F928,0084,A96E          SC_SC-K[.80]                 ;REMOVE BIAS FROM EXP
                                    =;END
                                            ;-------------------------------;
                                            D_D.LEFT, SI/DIVD, Q_0,         ; NORMALIZE MANTISSA<H> IN D, CLR Q
                                              ALU_0+Q, RC[T2]_ALU.LEFT,     ; SAVE NORMALIZED MANTISSA<L> IN RC[T2]
                                                 SC_SC-K[.20], SC?,         ; (PSL[N]=0 SO NO GARBAGE SHIFTS IN)
U 096E, 053F,1414,7478,F990,0084,A4D4                J/CVTFI.2              ; TURN EXP INTO SHIFT CT AND TEST RANGE
```

```
                                            .TOC    "       F & D floating point   : ACBF"

                                            ;GET HERE WITH LIMIT OPERAND IN Q, ADDEND IN D

                                            3C5:      ;------------------------------;
                                            ACBF:     R[R15]_D,                          ;SAVE ADDEND (SRC)
                                                      D_D(FRAC),SC_D(EXP),               ;UNPACK ADDEND
                                                      SS_ALU15,                          ;SETUP SRC SIGN FLAG
U 03C5, 0901,003C,1981,FAF8,1497,64A4                 STATE_K[ZERO],CLK.UBCC             ;CLEAR STATE AND EALU CC

                                            =0***00100;------------------------------;CALL SITE FOR GETTING INDEX
                                                      RC[T3]_Q,                          ;SAVE LIMIT
                                                      CHK.FLT.OPR,                       ;PROTECT AGAINST RESERVED OPERAND
                                                      ID[T2]_D,                          ;SAVE ADDEND FRAC WHILE GETTING BDEST
U 04A4, 0001,203D,C980,3D98,0800,037E                 CALL,J7SPEC                        ;GO GET INDEX OPERAND

                                            =0***10100;------------------------------;RETURN HERE WITH MEMORY OPERAND
U 04B4, 0000,003C,0580,F800,1404,64B6                 STATE_K[.1]                        ;FLAG INDEX AS MEMORY OPERAND

                                            =0***10110;------------------------------;RETURN HERE WITH REGISTER OPERAND
                                                      RC[T0]_D,                          ;SAVE INDEX (DST) OPERAND
                                                      D_D(FRAC),FE_D(EXP),               ;UNPACK INDEX OPERAND
                                                      CHK.FLT.OPR,CLK.UBCC,              ;CHECK RESERVED OPERAND
                                                      SS_SS.XOR.ALU15&SD_ALU15,          ;SETUP OP SELECT AND DEST SIGN
                                                      Q_IB.BDEST,PC_PC+2,                ;GET BRANCH DISPLACEMENT FROM IB
U 04B6, 7901,0B3D,01F5,F985,0918,6698                 CALL,IB.TEST?,J/ACBF.3

                                            =1***10110;------------------------------;RETURN HERE IF ADD UNNECESSARY
                                                      ALU_D.XOR.R[R15],SS_ALU15,         ;SET SS TO DIFF OF INDEX & ADDEND SIGNS
                                                      SC_K[ZERO],                        ;CLEAR OVERFLOW FLAG
U 05B6, 000D,1720,1981,FA78,0084,6598                 STATE0?,J/ACBF.5                   ;TEST WHERE TO STORE INDEX

                                            ; ***********************************************
                                            ; * Patch no. 009, PCS 05B6 trapped to WCS 1149 *
                                            ; ***********************************************

                                            =1***11110;------------------------------;NORMAL COMPLETION OF FLOATING ADD
                                            ACBF.2: EALU_SC,D_PACK.FP,                   ;REBUILD FLOATING RESULT
                                                      SET.CC(INST),                      ;SET CONDITION CODES ON IT
U 05BE, 0808,D438,0180,F800,0070,0691                 SC?,J/ACBF.4                       ;GO TEST FOR OVER/UNDERFLOW

                                            =1***11111;------------------------------;ROUNDING CAUSED DENORMALIZATION
                                                      D_D.RIGHT,                         ;RESTORE NORMALIZATION OF FRACTION
U 05BF, 0600,003C,0180,F800,0080,C5BE                 SC_SC+1,J/ACBF.2                   ;GO PACK UP RESULT
```

```
                                       ;HERE IN ACB FLOATING, TO GET THE BRANCH DESTINATION AND BEGIN THE ADD

                                       =00       ;-------------------------------;TB MISS
U 0698, 0000,003D,0180,F800,0000,0E6E  ACBF.3: CALL,J/IB.TBR                     ;TB REFILL REQUIRED

                                                 ;-------------------------------;ERROR
U 0699, 0000,003D,0180,F800,0000,0B80            CALL,J/IB.ERR

                                                 ;-------------------------------;STALL
                                                 Q_IB.BDEST,                     ;TRY AGAIN TO GET BRANCH DISPLACEMENT
U 069A, 7000,0B3C,01F0,F800,0000,0698            IB.TEST?,J/ACBF.3               ;LOOP UNTIL IT COMES

                                                 ;-------------------------------;GOT IT
                                                 RC[T7]_Q+PC,                    ;SAVE BRANCH DESTINATION
                                                 CLR.IB.SPEC,                    ;CLEAR 1ST BYTE OF BDEST
                                                 Q_ID[T2],                       ;GET BACK ADDEND FRACTION
                                                 SC_NABS(SC-FE),CLK.UBCC,        ;CALCULATE EXPONENT DIFFERENCE
U 069B, D015,3214,C9F0,2DB8,0090,E689            EALU?                           ;CHECK EXPONENTS FOR ZERO

                                       =;END OF CONSTRAINT FOR IB.TEST

                                       =1001     ;-------------------------------;EALU Z=0, SC.EQL.0
                                                 ALU_R[R15],CHK.FLT.OPR,         ;ADDEND EXP IS ZERO.  RESERVED OPERAND?
                                                 CLR.IB.SPEC,                    ;CLEAR 2ND BYTE OF BDEST
U 0689, D000,003C,0180,FA78,0800,0972            J/ACBF.3A                       ; IF NOT, RETURN INDEX AS SUM

                                                 ;-------------------------------;EALU Z=0, SC.NEQ.0
                                                 ID[T1]_D,                       ;SAVE INDEX FRACTION,
                                                 CLR.IB.SPEC,                    ;CLEAR 2ND BYTE OF BDEST
U 068B, D000,123C,C580,3C00,0000,05A2            EALU?,J/ADDFSH                  ;GO ADD FLOATING

                                                 ;-------------------------------;EALU Z=1, SC.EQL.0
                                                 D_0,                            ;BOTH ZERO, SUM IS ZERO
                                                 ALU_R[R15],CHK.FLT.OPR,         ;BEWARE RESERVED OPERAND
                                                 CLR.IB.SPEC,                    ;CLEAR 2ND BYTE OF BDEST
U 068D, DF00,003E,0180,FA78,0800,0100            RETURN100

                                                 ;-------------------------------;EALU Z=1, SC.NEQ.0
                                                 D_R[R15],                       ;INDEX IS ZERO, RETURN ADDEND AS SUM
                                                 CLR.IB.SPEC,                    ;CLEAR 2ND BYTE OF BDEST
U 068F, D800,003E,0180,FA78,0000,0100            RETURN100

                                       =;END OF CONSTRAINT FOR EXPONENT TEST

                                                 ;-------------------------------;
                                       ACBF.3A:D_RC[T0],                         ;ADDEND IS ZERO, RETURN INDEX UNCHANGED
U 0972, 0810,003A,0180,F900,0000,0100            RETURN100
```

```
                                         ;HERE IN ACBF, TO CHECK FOR UNDER/OVERFLOW OF EXPONENT

                                         =0001    ;0001---------------------------;SC.EQL.0
                                         ACBF.4: D_K[ZERO],SET.CC(INST),          ;RETURN ZERO ON UNDERFLOW
                                                 SC_K[ZERO],                      ;CLEAR OVERFLOW FLAG
U 0691, 0818,C039,1980,F800,00F4,6E09            CALL,J/UNDRFL

                                         FL.PA.9:;0011---------------------------;1.LEQ.SC.LEQ.FF
                                                 ALU_D.XOR.R[R15],SS_ALU15,       ;DIFF OF ADDEND & INDEX SIGNS TO SS
                                                 SC_R[ZERO],                      ;CLEAR OVERFLOW FLAG
U 0693, C00D,1720,1981,FA78,0084,6598            STATE0?,J/ACBF.5                 ;WHERE IS RESULT STORED?

                                                 ;0101---------------------------;SC.LSS.0
                                                 D_K[ZERO],SET.CC(INST),          ;RETURN ZERO ON UNDERFLOW
                                                 SC_K[ZERO],                      ;CLEAR OVERFLOW FLAG
U 0695, 0818,C039,1980,F800,00F4,6E09            CALL,J/UNDRFL

                                                 ;0111---------------------------;SC.GTR.FF
                                                 D_K[.8000],SET.CC(INST),         ;RETURN RESERVED OP ON OVERFLOW
U 0697, 0818,C039,4580,F800,0070,0E03            CALL,J/OVFL

                                         =1111    ;1111---------------------------;RETURN AFTER OVER/UNDERFLOW
                                                 ALU_R[R15],SS_ALU15,             ;ADDEND SIGN TO SS
U 069F, 0000,173C,0181,FA78,0000,0598            STATE0?

                                         =;END OF CONSTRAINT FOR OVER/UNDERFLOW TEST

                                         =0       ;-------------------------------;INDEX IS IN REGISTER
                                         ACBF.5: R(PRN)_D,LONG,                   ;STORE RESULT
                                                 Q_D,STATE_K[.1],                 ;SET "SINGLE" FLAG
U 0598, 0001,1A3C,05E0,F8D8,1404,659D            PSL.V?,J/ACBF.6

                                                 ;-------------------------------;INDEX IN MEMORY
                                                 CACHE_D[INST.DEP],               ;STORE IT
                                                 Q_D,STATE_K[.1],                 ;SET "SINGLE" FLAG
U 0599, 0000,DA3C,05E0,3000,1404,659D            PSL.V?,J/ACBF.6                  ;
```

```
                                          ;12465  ;HERE FOR ACBF OR ACBD AFTER STORING UPDATED INDEX.
                                          ;12466
                                          ;12467  =1101      ;---------------------------------;PSL.V =0
                                          ;12468  ACBF.6: D_Q.XOR.RC[T3],                     ;GET DIFFERENCE BETWEEN INDEX AND LIMIT
                                          ;12469          WORD,CLK.UBCC,                      ;TESTING ONLY SIGN, EXP, AND MSB'S
                                          ;12470          SD_SS,                              ;COPY XOR OF ADDEND & INDEX SIGNS TO SD
                                          ;12471          FE_K[.1],                           ;CLEAR EALU CC'S
U 059D, 0811,6020,0584,F918,0114,6974     ;12472          J/ACBF.6A
                                          ;12473
                                          ;12474  ; ************************************************
                                          ;12475  ; * Patch no. 001, PCS 059D trapped to WCS 1140 *
                                          ;12476  ; ************************************************
                                          ;12477
                                          ;12478          ;---------------------------------;PSL.V =1
U 059F, C000,003C,0180,F804,4000,0062     ;12479          CLR.IB.OPC,PC_PC+1,J/IRD            ;DO NOT BRANCH
                                          ;12480
                                          ;12481          ;---------------------------------;
                                          ;12482  ACBF.6A:D_D.OXT[WORD].XOR.Q,                ;NOW D<31:16>=INDEX, D<15:0>=LIMIT<15:0>
U 0974, 081F,5B20,0180,F800,0000,06A3     ;12483          ALU?                                ;TEST DIFFERENCE <15:0>
                                          ;12484
                                          ;12485  =0011      ;---------------------------------;ALU N&Z=0 (SIGNS SAME, MAGN DIFFER)
U 06A3, 001D,2020,018C,F800,0010,06A7     ;12486          ALU_Q.XOR.D,CLK.UBCC                ;COMPARE INDEX MAGNITUDE TO LIMIT
                                          ;12487
                                          ;12488          ;---------------------------------;ALU Z=1 (BITS 15:0 EQUAL)
                                          ;12489          ALU_Q.XOR.LC,LONG,CLK.UBCC,         ;SET Z IF EQUAL IN 32 BITS
                                          ;12490          EALU_FE,                            ;KEEP EALU CC CLEAR
                                          ;12491          SD_NOT.SD,                          ;SD=1 IFF ADDEND SIGN = INDEX SIGN
                                          ;12492          Q_ID[T6],                           ;GET LIMIT<L> IN CASE ACBD
U 06A7, 0011,2320,D9F3,2C00,0010,66B0     ;12493          C31?,J/ACBF.8                       ;TEST MAGNITUDE COMPARE
                                          ;12494
                                          ;12495          ;---------------------------------;ALU N=1 (SIGNS DIFFER)
                                          ;12496          CLR.IB.OPC,PC_PC+1,
U 06AB, C000,123C,0180,F804,4000,05A4     ;12497          EALU?                               ;TEST ADDEND SIGN .XOR. INDEX SIGN
                                          ;12498
                                          ;12499  =;END OF ALU N&Z CONSTRAINT
                                          ;12500
                                          ;12501  =0         ;---------------------------------;SS =0 DO NOT BRANCH
U 05A4, F80C,003B,01F1,F857,139B,6000     ;12502  ACBF.7: IRD
                                          ;12503
                                          ;12504          ;---------------------------------;SS =1 BRANCH
U 05A5, 2010,0038,0180,F939,4200,00AB     ;12505          PC&VA_RC[T7],FLUSH.IB,J/IB.FILL
```

```
                                            ;HERE IN ACBF/D FOR COMPARE OF INDEX TO LIMIT,
                                            ; WHEN BITS <15:0> OF INDEX AND LIMIT ARE EQUAL.

                                            =00        ;---------------------------------;ALU C=0 (INDEX .LEQ. LIMIT)
                                            ACBF.8:  SS_SD,                              ;GET SS= ADDEND SIGN.EQV.INDEX SIGN
                                                     D_Q.XOR.RC[T1],                     ;COMPARE LIMIT<L> TO INDEX<L>
                                                     WORD,CLK.UBCC,EALU_FE,
U 06B0, 0811,6120,0182,F908,0010,66B2               Z?                                  ;IF EQUAL, ALWAYS BRANCH

                                            =10        ;---------------------------------;ALU C=1 OR Z=0
                                                     CLR.IB.OPC,PC_PC+1,
U 06B2, C000,123C,0180,F804,4000,05A4               EALU?,J/ACBF.7

                                            =11        ;---------------------------------;Z=1 (`NDEX .EQL. LIMIT)
                                                     D_D.OXT[WORD].XOR.Q,                ;D<31:16>=LIMIT, D<15:0>=INDEX
U 06B3, 081F,5720,0180,F800,0000,05A8               STATE0?                             ;HAVE WE COMPARED FULL OPERANDS?

                                            =0         ;---------------------------------;STATE 0=0, MUST COMPARE LOW OF DOUBLE
                                                     ALU_Q.XOR.D,CLK.UBCC,LONG,          ;COMPARE LIMIT<47:32> WITH INDEX
                                                     EALU_FE,                            ;KEEP EALU CC'S CLEAR
                                                     SD_NOT.SD,                          ;SD= ADDEND SIGN.XOR.INDEX SIGN
U 05A8, 001D,3B20,0183,F800,0010,66BA               ALU?,J/ACBF.10

                                                     ;---------------------------------;STATE 0=1. INDEX .EQL. LIMIT
U 05A9, 2010,0038,0180,F939,4200,00AB               PC&VA_RC[T7],FLUSH.IB,J/IB.FILL ;BRANCH


                                            =1010      ;---------------------------------;ALU Z =0
U 06BA, 0000,033C,0180,F800,0000,06B0      ACBF.10:C31?,J/ACBF.8                        ;LIMIT<47:32> .NEQ. INDEX<47:32>

                                                     ;---------------------------------;ALU Z =0
U 06BB, 0000,033C,0180,F800,0000,0630               C31?,J/ACBF.8

                                                     ;---------------------------------;ALU Z =1, C31 =0 (LIMIT.LSS.INDEX)
U 06BE, 0000,003C,0182,F800,0000,06BF               SS_SD                               ;INVERT BRANCH SENSE YET AGAIN

                                                     ;---------------------------------;ALU Z =1, C31 =1 (LIMIT.GTR.INDEX)
                                                     CLR.IB.OPC,PC_PC+1,
U 06BF, C000,123C,0180,F804,4000,05A4               EALU?,J/ACBF.7
```

```
                                          .TOC    "       F & D floating point  : ACBD"

                                          ;HERE WITH LIMIT<H> IN RC[T0], LIMIT<L> IN Q,
                                          ; ADDEND<H> IN RC[T1], AND ADDEND<L> IN D

                                          384:        ;---------------------------------;
                                          ACBD:       RC[T6]_D(B),                       ;SAVE ADDEND<L> FOR UNPACK
                                                      D_Q,                               ;READY TO SAVE LIMIT<L>
                                                      SC_0(A),CLK.UBCC,                  ;INIT FOR EALU BRANCH
U 0384, 081F,2038,0180,F980,1498,6975                 STATE_0(A)                         ;INIT STATE

                                                      ;---------------------------------;
                                                      ID[T6]_D,                          ;SAVE LIMIT<L> TOO
                                                      D_RC[T0],                          ;READY TO SAVE LIMIT<H>
U 0975, 0810,0038,D980,3D00,4000,0050                 INTRPT.STROBE

                                          =10****0;---------------------------------;CALL SITE FOR GETTING INDEX
                                                      ID[T5]_D,                          ;SAVE LIMIT<H>
                                                      ALU_RC[T1],SS_ALU15,               ;GET ADDEND SIGN TO SS
U 0050, 0010,0E39,D581,3D08,0000,047E                 CALL,INTERRUPT.REQ?,J/ASPC         ;GO GET INDEX ADDRESS

                                          =11****0;---------------------------------;RETURN HERE WITH MEMORY OPERAND
                                                      ID[T7]_D,                          ;SAVE ADDRESS OF INDEX
U 0070, 0000,003C,DD80,3C00,0000,0978                 J/ACBD.2                           ;THEN GO GET INDEX

                                          =11****1;---------------------------------;HERE WITH REGISTER OPERAND
U 0071, 0001,003C,0180,F980,0000,0976                 RC[T0]_D                           ;SAVE INDEX<H>

                                                      ;---------------------------------;
                                                      D_R(PRN+1),                        ;GET INDEX<L>
U 0976, 0800,123C,0180,F860,0000,00B4                 EALU?,J/ACBD.4                     ;TEST ADDEND SIGN

                                          ;HERE WHEN INDEX IS IN MEMORY.  ADDRESS HAS BEEN SAVED IN ID[T7]

                                                      ;---------------------------------;
                                          ACBD.2:     D_CACHE.INST.DEP,                  ;GET INDEX<H>
U 0978, 0000,C03C,6580,5800,1404,697A                 STATE_K[.10]                       ;SET MEMORY OPERAND FLAG

                                                      ;---------------------------------;
                                                      RC[T0]_D,                          ;SAVE INDEX<H> WHERE UNPACK WILL FIND IT
U 097A, 0001,003C,0180,F983,0000,097C                 VA_VA+4                            ;GET ADDRESS FOR INDEX<L>

                                                      ;---------------------------------;
                                                      D[LONG]_CACHE,                     ;GET INDEX<L>
U 097C, 0000,123C,0180,4000,0000,00B4                 EALU?,J/ACBD.4                     ;TEST ADDEND SIGN
```

```
                                      ;HERE IN ACBD WITH INDEX<L> IN D, BRANCHING ON ADDEND SIGN

                                      =1*0     ;------------------------------;SS =0 (ADDEND IS POSITIVE)
                                      ACBD.4:  RC[T3]_D,                       ;SAVE INDEX<L> FOR UNPACK ROUTINE
                                               Q_IB.BDEST,PC_PC+1,             ;GET BRANCH DISPLACEMENT
U 00B4, 7001,0B3C,01F0,F99C,0000,06C0          IB.TEST?,J/ACBD.5               ;WAIT UNTIL IT ARRIVES

                                               ;------------------------------;SS =1 (ADDEND IS NEGATIVE)
                                               STATE_STATE.OR.K[.20],          ;REMEMBER THAT
                                               RC[T3]_D,                       ;SAVE INDEX<L> FOR UNPACK ROUTINE
                                               Q_IB.BDEST,PC_PC+1,             ;GET BRANCH DISPLACEMENT
U 00B5, 7001,0B3C,75F0,F99C,1404,26C0          IB.TEST?,J/ACBD.5               ;WAIT UNTIL IT ARRIVES

                                      =00      ;------------------------------;
U 06C0, 0000,003D,0180,F800,0000,0E64 ACBD.5:  CALL,J/IB.TBM                   ;REFILL TB

                                               ;------------------------------;
U 06C1, 0000,003D,0180,F800,0000,0880          CALL,J/IB.ERR                   ;SERVE ERROR

                                               ;------------------------------;STALL
U 06C2, 7000,0B3C,01F0,F800,0000,06C0          Q_IB.BDEST,IB.TEST?,J/ACBD.5    ;WAIT FOR BDEST TO ARRIVE

                                               ;------------------------------;GOT IT
                                               RC[T7]_Q+PC,                    ;CALCULATE BRANCH ADDRESS
                                               CLR.IB.SPEC,                    ;CLEAR 1ST BYTE OF BDEST
                                               Q_D,                            ;COPY INDEX<L> FOR SWAP
U 06C3, D015,2014,65E0,F9B8,0084,62C8          SC_K[.10]                       ;SETUP SC FOR SWAP OF HALVES

                                      ; ********************************************
                                      ; * Patch no. 003, PCS 06C3 trapped to WCS 1142 *
                                      ; ********************************************

                                      =0**00   ;------------------------------;CALL SITE FOR UNPACK
                                               D_DAL.SC,                       ;SWAP HALVES OF INDEX<L>
                                               SC_K[.FFF9],                    ;GET -7 FOR SHIFT
U 02C8, DD00,003D,BD80,F800,0084,66CA          CLR.IB.SPEC,CALL,J/UNPACK       ;CLEAR 2ND BYTE OF BDEST

                                               ;------------------------------;SRC.EQL. 0
                                               Q_ID[T7],                       ;GET ADDRESS OF INDEX IF MEMORY
U 02C9, 0000,003C,DDF0,2D08,0000,02CA          LC_RC[T1]                       ;GET INDEX<L> TO LATCH
U 02CA, 0000,163C,0180,F800,0000,06D2          STATE4?,J/ACBD.6                ;WHERE SHOULD RESULT BE STORED?

                                               ;------------------------------;DST.EQL.0
                                               Q_ID[T7],                       ;GET ADDRESS OF INDEX IF MEMORY
U 02CB, 0000,003C,DDF0,2D08,0000,02D8          LC_RC[T1]                       ;GET INDEX<L> TO LATCH
U 02D8, 0000,163C,0180,F800,0000,06D2          STATE4?,J/ACBD.6                ;WHERE SHOULD RESULT BE STORED?

                                               ;------------------------------;NEITHER ZERO
                                               FE_NABS(SC-LA(EXP)),            ;CALCULATE SHIFT AMOUNT
U 02D9, 0000,003D,0180,F800,0118,E583          CLR.UBCC,CALL,J/ADDD.6          ;NOTE ITS DIRECTION, GO FINISH THE ADD

                                      =1**11   ;------------------------------;RETURN FROM ADDD/PACKD
                                               Q_ID[T7],                       ;GET ADDRESS OF INDEX IF MEMORY
U 02DB, 0000,003C,DDF0,2D08,0000,097D          LC_RC[T1]                       ;GET INDEX<L> TO LATCH
U 097D, 0000,163C,0180,F800,0000,06D2          STATE4?,J/ACBD.6                ;WHERE SHOULD RESULT BE STORED?
```

```
                                                  ;HERE IN ACBD TO STORE INDEX

                                                  =10      ;-------------------------------;STATE 4=0 (INDEX IN REGISTER)
                                                  ACBD.6:  R(PRN)_D,SS_ALU15,              ;STORE INDEX<H>, SET SS FROM SIGN
                                                           ID[T0]_D,                       ;SAVE INDEX<H> WHILE <L> BEING STORED
U 06D2, 0001,003C,C181,3CD8,0000,0982                      J/ACBD.7                        ;TEST FOR NEGATIVE ADDEND

                                                           ;-------------------------------;STATE 4=1 (INDEX IN MEMORY)
                                                           VA_Q,                           ;RELOAD ADDRESS OF INDEX
U 06D3, 0001,203C,C180,3C00,0200,097E                      ID[T0]_D                        ;SAVE INDEX<H> DURING STORE

                                                           ;-------------------------------;
                                                           CACHE_D[INST.DEP],              ;STORE INDEX<H>
U 097E, 0001,C03C,0181,3000,0000,0980                      ALU_D,SS_ALU15                  ;SET SS FROM SIGN OF INDEX

                                                           ;-------------------------------;
                                                           VA_VA+4,                        ;ADVANCE ADDRESS TO INDEX<L>
                                                           D_[C,                           ;GET INDEX<L> FROM LATCH
U 0980, 0810,0038,D5F0,2C03,0000,0981                      Q_ID[T5]                        ;GET LIMIT<H> FOR COMPARE

                                                           ;-------------------------------;
                                                           CACHE_D[LONG],                  ;STORE INDEX<L>
U 0981, 0000,163C,0180,3000,0000,04E1                      STATE5?,J/ACBD.8

                                                           ;-------------------------------;
                                                  ACBD.7:  R(PRN+1)_LC,                    ;STORE INDEX<L>
                                                           Q_ID[T5],                       ;GET LIMIT<H> FOR COMPARE
U 0982, 0010,1638,D5F0,2CE0,0000,04E1                      STATE5?                         ;TEST FOR NEGATIVE ADDEND

                                                  =01      ;-------------------------------;STATE 5=0. (ADDEND IS POSITIVE)
                                                  ACBD.8:  RC[T3]_Q,                       ;PUT LIMIT WHERE ACBF WANTS IT
                                                           Q_ID[T0],                       ;GET INDEX TOO
U 04E1, 0001,3A3C,C1F0,2D98,0000,059D                      PSL.V?,J/ACBF.6                 ;GO COMPARE THEM

                                                           ;-------------------------------;STATE 5=1.  (ADDEND IS NEGATIVE)
                                                           ALU_K[.8000],                   ;SET ALU15
                                                           SS_SS.XOR.ALU15&SD_ALU15,       ;COMPLEMENT INDEX SIGN IN SS
U 04E3, 0018,0038,4585,F800,0000,04E1                      J/ACBD.8                        ;
```

```
                                      .TOC     "       F & D floating point   : MULD"

                                      .FFLIST          ;Re-enable full listing
                                      .REGION/<WCSR1L>,<WCSR1H>/<WCSR2L>,<WCSR2H>

                                      ;DOUBLE FLOATING POINT ARITHMETIC MULD ROUTINE.
                                      ;
                                      ;  USED BY POLYD AND EMODD AS WELL AS BY MULD
                                      ;
                                      ;  THIS ROUTINE MULTIPLIES A 56-BIT "DST" BY A 64 BIT "SRC"
                                      ;  TO PRODUCE A PRODUCT WITH 63 OR 64 SIGNIFICANT BITS, DEPENDING
                                      ;  ON THE MAGNITUDE OF THE INPUT FRACTIONS.
                                      ;
                                      ;  THE "DST" IS IN <RC1,D>; THE "SRC" IN <RC0,Q>
                                      ;
                                      ;  WHEN CALLED BY MULD OR POLYD, THE SRC REALLY HAS ONLY 56 SIGNIFICANT BITS.
                                      ;
                                      ;  THIS ROUTINE RETURNS 10 IF THE PRODUCT IS ZERO WITH D=Q=RC[T1]=SC=0.
                                      ;  IF THE PRODUCT IS NON-ZERO AND IT WAS CALLED FROM POLY OR EMOD
                                      ;  IT RETURNS 12 OR 13 WITH THE UNNORMALIZED PRODUCT IN <D,Q>, DEPENDING
                                      ;  ON HOW MANY LEADING ZERO BITS ARE IN THE PRODUCT.
                                      ;  SS AND SD BOTH HAVE THE RESULT SIGN
                                      ;  IF THE PRODUCT IS NON-ZERO AND IT WAS CALLED FROM MULD IT RETURNS
                                      ;  THE PACKED RESULT IN <D,RC[T1]>.
                                      ;  OVERFLOW/UNDERFLOW CHECKING IS ONLY DONE ON THE MULD PATH.


                                      1003:    ;ASSIGN THIS ADDRESS BECAUSE PCS CALLS IT
                                      MULD.00:;---------------------------;
                                               RC[T6]_D, D_Q,             ;   SAVE DSTO <L>, D GETS SRCO<L>
U 1003, 0C01,003C,6580,F9B0,0084,72C8          SC_K[.10]                  ;   SC GETS 16. FOR SWAP WORD OF FRAC <L>

                                      =00      ;00------------------------;
                                      MULD.02:RC[T3]_D, D_DAL.SC,         ;   SAVE SRCO <L>, SWAP WORDS OF SRCO<L>
                                               SC_K[.FFF9],               ;   SETUP SHIFT AMOUNT -7
U 12C8, 0D01,003D,BD80,F998,0084,66CA          CALL,J/UNPACK              ;   CALL UNPACK DOUBLE FLOATING PT OPERANDS ROUTINE

                                               ;01------------------------;   RETURN1, SRC = 0, DST MAY BE 0
                                               RC[T1]_0,D_0,Q_0,          ;   RESULT IS 0
                                               SC_ALU, FE_K[.10],         ;   CLR EXP FOR POLYD, SET FE FOR EMODD
U 12C9, 0F03,003E,65F8,F988,01D6,6010          N&Z_ALU.V&C_0, RETURN10    ;   SET CC'S, GOTO SET WRITE RESULT READY

                                               ;10------------------------;   RETURN2, SRC.NE.0, DST = 0
                                               RC[T1]_0,D_0,Q_0,          ;   RESULT IS 0
                                               SC_ALU, FE_K[.10],         ;   CLR EXP FOR POLYD, SET FE FOR EMODD
U 12CA, 0F03,003E,65F8,F988,01D6,6010          N&Z_ALU.V&C_0, RETURN10    ;   SET CC'S, GOTO SET WRITE RESULT READY

                                               ;11------------------------;   RETURN3, SRC.NE.0, DST.NE.0
                                               RC[T2]_D,                  ;   SAVE DST FRAC <L>, SET AS MULT'CAND*2
                                               D_D.RIGHT,SI/ZERO,         ;   DST FRAC <L>/2 AS MULT'CAND
                                               SC_FE,                     ;   SC GETS DST(EXP) - SRC(EXP)
U 12CB, 0601,173C,0184,F990,0081,12A9          SD_SS, STATE1?             ;   SET RESULT SIGN TO SD & CHECK IF EMODD
                                      =;END
```

```
                                  ;          AT THIS POINT,
                                  ;          ID[T0]=SRC<H>    ID[T1]=DST<H>    ID[T2]=SRC<L>
                                  ;          RC[T2]=DST<L>    RC[T5]=DST<H>    D      =DST<L>/2     Q  =SRC<L>

                                             ;----------------------;
                                  =**01      ;BRANCH ON STATE<1> (EMODD)
                                             ;0---------------------;  MULD (OR POLYD)
                                             R[R15]_D,CLK.UBCC,     ;  MULT'CAND TO R15
                                             D_Q, Q_0,              ;  MULTIPLIER TO D
U 12A9, 0C01,003C,41F8,FAF6,0114,B108        FE_SC-K[.80], J/MULD.03 ;  ADD EXP BIAS 128. TO TEMP EXP RESULT

                                             ;1---------------------;  EMODD - MUST ADD EXTENSION TO SRC FRACT
                                             Q_ID[T0], D_Q,         ;  SRC<H> IN Q, SRC<L> IN D
U 12AB, 0C01,003C,C1F0,2EF8,0010,137A          R[R15]_D, CLK.UBCC   ;  SAVE DST<L> WHILE WE PAD OUT SRC

                                             ;----------------------;
                                             D_D.LEFT, Q_Q.LEFT,    ;  SHIFT SRC FRACT LEFT
U 137A, 0500,003C,4128,F800,0104,B380          SI/ASHL, FE_SC-K[.80] ;  REMOVE EXTRA EXP BIAS

                                             ;----------------------;
U 1380, 0811,0030,0180,F920,0000,1384        D_D.OR.RC[T4]          ;  PUT SRC EXTENDER AT LOW END OF FRACT

                                             ;----------------------;
U 1384, 0C00,003C,C9E0,3C00,0000,1388        ID[T2]_D, D_Q, Q_D     ;  SAVE EXTENDED SRC IN ITS OLD SPOT

                                             ;----------------------;
U 1388, 0C00,003C,C1F8,3C00,0000,1108        ID[T0]_D, D_Q, Q_0     ;  FINISH SAVING SRC AND PREPARE TO MULTIPLY

                                  =0*        ;0---------------------;  DST <L> TIMES SRC <L>
                                  MULD.03:   LC_RC[T2],             ;  LATCH M'CAND * 2
                                             SC_K[.F],              ;  SET LOOP CT FOR 16. LOOPS
U 1108, 0000,013D,6180,F910,0084,70E8        CALL, Z?, J/MULDMPY    ;  GOTO MULTIPLICATION SETUP

                                             ;1---------------------;  RETURN FROM MUL SUBRT
                                                                    ;  PRODUCT IS 1 PLACE TOO FAR RIGHT, SINCE WE
                                             ALU_D, N_AMX.Z_TST,    ;  HALVED THE LOW DEST FRACT BEFORE MULTIPLYING;
                                               D_Q, Q_ID[T0],       ;  THEREFORE SAVE THE 32ND BIT IN PSL<N>.
U 110A, 0C01,003C,C1F0,2D10,0030,110C            LC_RC[T2]          ;  D GETS PROD<H>, BRING SRC<H> INTO Q.
                                  =;END

                                  =0*        ;0---------------------;  DST <L> TIMES SRC <H>
                                             D_Q, Q_D, SC_K[.F],    ;  IMMEDIATE PROD TO Q, MULT'IER TO D
U 110C, 0C00,013D,61E0,F800,0084,70E8        CALL, Z?, J/MULDMPY    ;  GOTO MULTIPLICATION SETUP

                                             ;1---------------------;  RETURN FROM MULTIPLY SUBROUTINE
                                                                    ;  THIS PRODUCT IS ALSO 1 PLACE TOO FAR RIGHT.
U 110E, 0C00,003C,01E0,F928,0000,1389        D_Q, Q_D, LC_RC[T5]    ;  PREPARE FOR LEFT SHIFT, LATCH DST<H>

                                             ;----------------------;
                                             ALU_Q, RC[T6]_ALU.LEFT, ;  SHIFT THE QUANTITY <D,Q,PSL<N>> LEFT 1 PLACE
U 1389, 0521,203C,0000,F9B0,0000,1390          D_D.LEFT, SI/DIVD    ;  TO FORM THE HIGH 64-BITS OF DST<L> X SRC
                                                                    ;  IN <D, RC6>
                                             ;----------------------;
                                             R[R15]_LC, ID[T3]_D,   ;  GET M'CAND, SAVE PROD<H> FOR LATER ADD
U 1390, 0010,0038,CDC0,3EF8,0000,1392          Q_LC                 ;
```

```
                                    ;12791              ;---------------------------;
                                    ;12792              RC[T0]_Q.LEFT,SI/ZERO,      ;   GET 2 TIMES M'CAND
U 1392, 0021,203C,C9F0,2D80,0000,1118 ;12793            Q_ID[T2]                    ;   GET SRC FRAC <L>
                                    ;12794
                                    ;12795   =0*        ;0--------------------------;   DST <H> TIMES SRC <L> PLUS (DST<L> X SRC)<L>
                                    ;12796              D_Q, Q_0, SC_K[.F],         ;   D GETS MULT'IER, Q GETS 0(CAN'T ADD - OVFLO)
                                    ;12797              LC_RC[T0],                  ;   LATCH 2 TIMES M'CAND
U 1118, 0C00,003D,61F8,F900,0084,70E8 ;12798            CALL, J/MULDMPY             ;   GOTO SETUP MULTIPLICATION
                                    ;12799
                                    ;12800              ;1--------------------------;   RETURN FROM MUL SUBRT
U 111A, 0811,0014,0180,F800,0010,1394 ;12801            D_D+LC, CLK.UBCC            ;   ADD (DST<L> X SRC)<L> TO (DST<H> X SRC<L>)<L>
                                    ;12802                                          ;   (LC CONTAINS RC[T6] FROM MULDMPY)
                                    ;12803   =;END
                                    ;12804
                                    ;12805              ;---------------------------;
                                    ;12806              ALU_D, N_AMX.Z_TST,         ;   SAVE CURRENT SUM<L><31> IN PSL<N>
U 1394, 0C01,033C,CDF0,2C00,0030,111C ;12807            D_Q, Q_ID[T3], C31?         ;   SET UP FOR HIGH-ORDER ADD
                                    ;12808
                                    ;12809   =0*        ;0--------------------------;
                                    ;12810              ALU_D+Q, RC[T6]_ALU,        ;   NO CARRY - ADD, SAVE RESULT ACROSS FINAL MPY
                                    ;12811              CLK.UBCC,                   ;  SAVE THE CARRY IF THER IS ONE
U 111C, 001D,0014,C1F0,2DB0,0010,1130 ;12812              Q_ID[T0], J/MULD.04       ;   LOAD SRC<H> FOR FINAL MULTIPLY
                                    ;12813
                                    ;12814              ;1--------------------------;
                                    ;12815              ALU_D+Q+1, RC[T6]_ALU,      ;   CARRY - ADD WITH CARRY, SAVE RESULT,
                                    ;12816              CLK.UBCC,                   ;  SAVE THIS CARRY IF THERE IS ONE
U 111E, 001D,0010,C1F0,2DB0,0010,1130 ;12817              Q_ID[T0]                  ;   Q = SRC<H> FOR LAST MULTIPLY
                                    ;12818   =;END
                                    ;12819
                                    ;12820   =0*        ;0--------------------------;   FRAC <H> TIMES FRAC <H>
                                    ;12821   MULD.04:   LC_RC[T0],                  ;   RE-LATCH 2 TIMES M'CAND
                                    ;12822              D_Q, Q_0,                   ;   MULT'IER <H> TO D, Q = 0 (OVFLO PROBLEMS)
                                    ;12823              SC_K[.F],                   ;   SETUP LOOP CT
U 1130, 0C00,003D,61F8,F900,0084,70E8 ;12824            CALL, J/MULDMPY             ;   GOTO SETUP MULTIPLICATION
                                    ;12825
                                    ;12826              ;1--------------------------;   RETURN FROM MUL SUBRT
                                    ;12827              Q_D+LC, CLK.UBCC, D_Q,      ;   SWAP HALVES, ADDING IN OLD PARTIAL PRODUCT
U 1132, 0C11,0314,01C0,F800,0010,1134 ;12828            C31?                        ;  BRANCH ON CARRY FROM PREVIOUS ADD
                                    ;12829   =;END
                                    ;12830
                                    ;12831   =0*        ;---------------------------;
U 1134, 0000,033C,0180,F800,0000,113C ;12832            C31?,J/MULD.05              ;   CHECK FOR CARRY INTO HIGH PRODUCT
                                    ;12833              ;---------------------------;
                                    ;12834              D_D+K[.1],                  ;
U 1136, 0819,0314,0580,F800,0000,113C ;12835            C31?                        ;  PROPOGATE THE CARRY
                                    ;12836
                                    ;12837   =0*        ;0--------------------------;   NO CARRY
                                    ;12838   MULD.05:   ALU_Q, Q_ALU.LEFT,          ;   SHIFT <D,Q,PSL<N>> LEFT ONE
                                    ;12839              D_D.LEFT, SI/DIVD,          ;   TO FORM 64-BIT PRODUCT IN <D,Q>
                                    ;12840              STATE_STATE.ANDNOT.SHF.VAL, ; CLR STATE<0> IF FRACT<.5 (POLYD ONLY)
U 113C, 0521,373C,0040,F800,148D,5262 ;12841            SC_FE, STATE0?, J/MULD.06   ;  SC=EXP, SEPARATE OUT MULD FROM EMODD/POLYD
                                    ;12842
                                    ;12843              ;1--------------------------;   CARRY FROM ADD
U 113E, 0819,0014,0580,F800,0000,113C ;12844            D_D+K[.1], J/MULD.05        ;   PROPAGATE IT AND CONTINUE
                                    ;12845   =;END
```

```
                                         ;EXIT FROM MULD FOR ALL USERS (MULD, EMODD, POLYD)

                                         =**10   ;0---------------------;  THIS IS MULD
                                         MULD.06: Q_Q.LEFT, D_D.LEFT,   ;  WE WANT <D,Q> NORMALIZED FOR
                                                 SI/DIVD,               ;  ROUNDING - THE FIRST SHIFT IS FREE.
U 1262, 0500,003C,7C28,F800,0104,7396           FE_K[.18], J/MULD.07   ;  SET UP FE FOR PACKD, GO TEST IF NORM

                                                 ;1---------------------;  POLYD/EMODD
U 1263, 0000,173E,0180,F800,0000,0012           STATE0?, RETURN12      ;  TEST NORMALIZATION IN CASE ITS POLYD
                                         =;END


                                                 ;----------------------;
U 1396, 0001,0D3C,4180,FAF8,0000,12C6   MULD.07: R[R15]_D, K[.80], D31? ;  SAVE PROD<H>, CHECK IF NORMALIZED

                                                 ;----------------------;
                                         =110    ;BRANCH ON D31 (PRODUCT NORMALIZED)
                                                 ;0---------------------;  D31 IS 0
                                                 D_D.LEFT,Q_Q.LEFT,     ;  DOUBLE SHIFT PROD LEFT
                                                 SI/DIVD,               ;
                                                 SC_SC-K[.1],           ;  DECREMENT EXP TO COMPENSATE FOR SHIFT
U 12C6, 0500,003C,0428,F800,0084,B396           J/MULD.07              ;  TEST AGAIN (GUARANTEED TO SUCCEED 2ND TIME)

                                                 ;1---------------------;
                                                 Q_Q+K[.80], CLK.UBCC,  ;  ROUND PROD FRAC <L>, SET C31 FOR ROUNDING
U 12C7, 0019,2014,41C0,F800,0010,03FC           J/PACKD                ;  GO TO PACK RESULT
                                         =;END
                                         ;       FRACTION MULTIPLY ROUTINE FOR MULTIPLY DOUBLE - MULTIPLIES
                                         ;       AN UNSIGNED 32-BIT MULTIPLIER BY AN UNSIGNED 31-BIT MULTIPLICAND.
                                         ;       LOGIC IS  V E R Y  PARALLEL TO INTEGER MULTIPLY - SEE COMMENTS THERE.

                                         =0                             ;  ENTERED WITH "Z?"  TEST FOR M'CAND=0
                                         MULDMPY:;0---------------------;  MULTIPLICAND NOT ZERO - SETUP FOR LOOP
                                                 ALU_O(A), D_D.RIGHT2,  ;  D GETS M'CAND
                                                 LAB_R[R15], SI/ZERO,   ;  LATCH M'CAND TO LB
U 10E8, 0203,0C3C,6180,FA78,0084,72FC           SC_K[.F], MUL?,J/MULPAP ;  SETUP LOOPCOUNT & GOTO MULTIPLICATION ROUTINE

                                                 ;1---------------------;  M'CAND IS 0
                                                 D_0, Q_0, ALU_O(A),    ;
                                                 LAB_R[R15],            ;  LATCH M'CAND TO LB ANYWAY
U 10E9, 0F03,003C,01F8,FA78,0000,12F0           J/MULPAP               ;  LATCH RC[T6] AND EXIT
                                         =;END
```

```
                                        ;12887  ;MULTIPLY LOOP HERE - ENTER VIA   'MUL?, J/MULPAP''
                                        ;12888
                                        ;12889  =000
U 12F0, 0200,003E,0300,F930,4000,0002   ;12890  MULPAP: LC_RC[T6], MULP.DONE, RETURN2          ;RETURN TO RETURN ADDR .OR. 2
                                        ;12891
                                        ;12892  =100
U 12F4, 0281,2C3C,0740,F800,0084,B2F0   ;12893          QD_QD.RIGHT2,       MUL.0XT, J/MULPAP    ;+0, 0XT
U 12F5, 028D,2C14,0740,F800,0084,B2F0   ;12894          QD_(Q+LB)D.RIGHT2, MUL.0XT, J/MULPAP    ;+1, 0XT
U 12F6, 0291,2C00,07C0,F800,0084,B320   ;12895          QD_(Q-LC)D.RIGHT2, MUL.1XT, J/MULPAM    ;-2, 1XT
U 12F7, 028D,2C00,07C0,F800,0084,B320   ;12896          QD_(Q-LB)D.RIGHT2, MUL.1XT, J/MULPAM    ;-1, 1XT
                                        ;12897
                                        ;12898
                                        ;12899  =000
                                        ;12900  MULPAM: LC_RC[T6], ALU_Q+LB, Q_ALU,          ;RETURN TO RETURN ADDR .OR. 2
U 1320, 020D,2016,0340,F930,4000,0002   ;12901          MULP.DONE, RETURN2                  ;AFTER CORRECTING PRODUCT
                                        ;12902
                                        ;12903  =100
U 1324, 028D,2C14,0740,F800,0084,B2F0   ;12904          QD_(Q+LB)D.RIGHT2, MUL.0XT, J/MULPAP    ;+1, 0XT
U 1325, 0291,2C14,0740,F800,0084,B2F0   ;12905          QD_(Q+LC)D.RIGHT2, MUL.0XT, J/MULPAP    ;+2, 0XT
U 1326, 028D,2C00,07C0,F800,0084,B320   ;12906          QD_(Q-LB)D.RIGHT2, MUL.1XT, J/MULPAM    ;-1, 1XT
U 1327, 0281,2C3C,07C0,F800,0084,B320   ;12907          QD_QD.RIGHT2,       MUL.1XT, J/MULPAM    ;-0, 1XT
```

```
                                        ;12908  .TOC    "       F & D floating point   : EMODF"
                                        ;12909
                                        ;12910  ;EMODF  (54)    MULR.RF, MULRX.RB, MULD.RF, INT.WL, FRACT.WF
                                        ;12911  ;ENTER WITH Q = MULR.RF, D = MULRX.RB AT C.FORK.
                                        ;12912
                                        ;12913  3C8:
                                        ;12914  EMODF:  ;---------------------------------------;
                                        ;12915          Q_D,                                    ; MOVE EXT'ER TO Q
                                        ;12916          D_Q(FRAC),                              ; MUL'IER FRAC
                                        ;12917          SC_Q(EXP),                              ; MUL'IER EXP
                                        ;12918          SS_ALU15,                               ; MUL'IER SIGN
U 03C8, 0901,203C,01E1,F800,0888,7023   ;12919          CHK.FLT.OPR,J/FL.ABS.1023               ; CHECK FOR -0
                                        ;12920
                                        ;12921  1023:                                          ;ASSIGN THIS ADD BECAUSE PCS CALLS IT
                                        ;12922  FL.ABS.1023:
                                        ;12923          ;0**1*----------------------------------;
                                        ;12924          D_D.LEFT.SI/ZERO,                       ;  MOVE TO LEAVE ROOM FOR M'IER EXT'ER
                                        ;12925          RC[T1]_Q.OXT[BYTE],                     ;  RC 1 GETS M'IER EXT'ER (BYTE)
U 1023, 0503,AE3D,0180,F988,0000,037E   ;12926          CALL,INTERRUPT.REQ?,J/SPEC              ;  GO GET M'CAND
                                        ;12927
                                        ;12928  1033:   ;ASSIGN THIS ADDRESS BECAUSE OF CONSTRAINT ON PREVIOUS INSTRUCTION
                                        ;12929          ;1**1*----------------------------------;  RETURN FROM "SPEC"
                                        ;12930          D_D(FRAC),                              ;  GET M'CAND FRAC
                                        ;12931          FE_D(EXP),CLK.UBCC,                     ;  M'CAND EXP
                                        ;12932          LC_RC[T1],                              ;  LATCH UP MULTIPLIER EXTENDER
                                        ;12933          SS_SS.XOR.ALU15&SD_ALU15,               ;  SS GETS RESULT SIGN
U 1033, 0901,003C,0185,F908,0918,739D   ;12934          CHK.FLT.OPR                             ;  CHECK FOR -0
                                        ;12935  =;END
                                        ;12936          ;---------------------------------------;
                                        ;12937          R[R15]_D,                               ;  R15 GETS M'CAND
                                        ;12938          SC_SC+FE,                               ;  SC GETS SUM OF EXP'S
U 139D, 0001,123C,0180,FAF8,0080,9299   ;12939          EALU?                                   ;  M'IER OR M'CAND = 0?
                                        ;12940
                                        ;12941  =1001   ;1001-----------------------------------;  PROD = 0 (M'IER IS 0)
                                        ;12942          SC_K[ZERO],                             ;
                                        ;12943          RC[T1]_K[ZERO],                         ;  PROD SET TO 0
                                        ;12944          D_0,Q_0,SET.CC(INST),                   ;  PROD SET TO 0
U 1299, 0F18,C038,19F8,F988,00F4,703E   ;12945          J/EMODF.7                               ;  GOTO WRITE RESULT
                                        ;12946
                                        ;12947          ;1011-----------------------------------;  PROD .NE. 0
                                        ;12948          RC[T0]_D.LEFT,SI/ZERO,                  ;  RC 0 GETS M'CAND * 2
U 129B, 0021,003C,4180,F980,0084,B3A4   ;12949          SC_SC-K[.80],J/EMODF.2                  ;  SAVE EXP WITH BIAS ADJUSTED
                                        ;12950
                                        ;12951          ;1101-----------------------------------;  PROD = 0 (M'IER, M'CAND ARE 0)
                                        ;12952          SC_K[ZERO],                             ;
                                        ;12953          RC[T1]_K[ZERO],                         ;  PROD SET TO 0
                                        ;12954          D_0,Q_0,SET.CC(INST),                   ;  PROD SET TO 0
U 129D, 0F18,C038,19F8,F988,00F4,703E   ;12955          J/EMODF.7                               ;  GOTO WRITE RESULT
                                        ;12956
                                        ;12957          ;1111-----------------------------------;  PROD = 0 (M'CAND IS 0)
                                        ;12958          SC_K[ZERO],                             ;
                                        ;12959          RC[T1]_K[ZERO],                         ;  PROD SET TO 0
                                        ;12960          D_0,Q_0,SET.CC(INST),                   ;  PROD SET TO 0
U 129F, 0F18,C038,19F8,F988,00F4,703E   ;12961          J/EMODF.7                               ;  GOTO WRITE RESULT
```

```
                                              EMODF.2:;---------------------------------;
                                                      LAB_R[R15], SD_SS,                : LATCH M'CAND
                                                      ALU_Q.OR.LC, D_ALU,               : D GETS FULL 32 BIT MULTIPLIER
U 13A4, 0811,2030,41FC,FA78,0104,B20C                 Q_0, FE_SC-K[.80]                 : UNBIAS PRODUCT EXPONENT TO FE

                                              ;       A NOTE ON THE MULTIPLICATION:
                                              ;       THE MULTIPLICAND HAS BIT 31=0, BIT 30=1
                                              ;       THE MULTIPLIER HAS BIT 31 = 1, ALL BITS ARE SIGNIFICANT
                                              ;       THE MULTIPLICATION IS CARRIED TO 66 BITS (17 ITERATIONS) SO THE LAST
                                              ;       ITERATION WILL THINK THE MULTIPLIER IS POSITIVE.
                                              ;       THEREFORE THE PRODUCT WILL BE SHIFTED 3 BITS RIGHT FROM WHAT IT
                                              ;       WOULD HAVE BEEN IF BOTH FRACTIONS HAD BEEN NORMALIZED AND THE
                                              ;       MULTIPLICATION CARRIED TO 64 BITS.

                                              =0*     ;0*-------------------------------;
                                                      D_D.RIGHT2.SI/ZERO,               : SHF'G FOR MULIPLICATION
                                                      LC_RC[T0],                        : LATCH M'CAND * 2
                                                      SC_K[.10], ALU_Q(A),              : SETUP LOOP COUNT AND ALUO-1 LATCHES
U 120C, 0203,0C3D,6580,F900,0084,6354                 CALL,MUL?,J/MULPP.4               : CALL MUL SUBRT, GUARDING AGAINST SC=0

                                                      ;1*-------------------------------: RETURN
                                                      RC[T1]_0, N&Z_ALU.V&C_0,          : SET UP RC[T1] AND CC'S FOR CVTFI
                                                      SC_K[.3],                         : SET UP TO SCALE PRODUCT
U 120E, 0C03,003C,0DE0,F988,00D4,73A5                 D_Q,Q_D                           : SWAP SO D GETS PROD<H>, Q PROD<L>

                                                      ;---------------------------------: SCALE PROD SO D HAS
U 13A5, 0D00,003C,01F8,F800,0081,102E                 SC_FE, D_DAL.SC, Q_0              : 31/32 SIGNIFICANT BITS

                                              ; *****  ENTRY POINT FOR FPA *****
                                              ; D HAS FRAC, SC HAS UNBIASED EXP, SS&SD HAVE RESULT SIGN, RC[T1]=0, Z=1
                                              =0**1*
                                              EMODF.6:;0**1*----------------------------;
                                                      SC_SC-SHF.VAL,FE_EALU,            : NORMALIZE
                                                      D_DAL.NORM,                       :
U 102E, 0E00,003D,0180,F800,018C,A946                 CALL, J/CVTFI.1                   :

                                              EMODF.7:
                                                      ;1**1*----------------------------;
                                                      ID[T0]_D, Q_0, D_RC[T1],          : SET UP TO SHIFT FRACTION
U 103E, 0810,1438,C1F8,3D08,0000,1305                 SC?                               : BUT DON'T SHIFT IF NUM<1.0

                                              =101    ;101------------------------------: BRANCH ON SC (NUM => 1.0)
U 1305, 0000,003C,4180,F800,0084,93A6                 SC_SC+K[.80], J/EMODF.8           : NUM<1.0 - RE-BIAS EXPONENT

                                                      ;111------------------------------;
U 1307, 0D00,003C,4180,F800,0084,73A6                 SC_K[.80], D_DAL.SC               : NUM => 1.0 - THROW AWAY INTEGER
```

```
                                       EMODF.8:
                                                ;-------------------------------;
                                                SC_SC-SHF.VAL,                  ; NORMALIZE FRACTION
                                                D_DAL.NORM, K[.80],             ; SET UP ROUND CONSTANT
U 13A6, 0E00,0D3C,4180,F800,008C,82D9           D.NE.0?                         ; TEST FOR 0 FRACTION
                                       =*01
                                       EMODF.9:
                                                ;*01----------------------------; D=0
                                                ID[T1]_D, SC_0(A),              ; MAKE FRACTION A TRUE ZERO
                                                STATE_0(A),                     ; CLEAR STATE FOR DEST STORE FLAGS
                                                SGN/C[R.SD+SS,                  ;
U 12D9, 0003,003C,C587,3C00,1488,72A7           J/EMODFD                        ; GO STORE THE RESULTS

                                                ;*11----------------------------;
U 12DB, 0819,0014,4180,F800,0010,13AC           D_D+K[.80], CLK.UBCC            ; ROUND FRACTION

                                                ;-------------------------------;
U 13AC, 0000,033C,0180,F800,0100,D210           FE_SC+1, C31?                   ; TEST FRACTION OVERFLOW FOM ROUND

                                       =0*      ;0*-----------------------------;
                                                EALU_SC, D_PACK.FP,             ; NO OVERFLOW - USE UNINCREMENTED EXP
U 1210, 0808,0038,0180,F800,0000,13AD           J/EMODF.10                      ;

                                                ;1*-----------------------------;
                                                EALU_FE, SC_FE,                 ; OVERFLOW - USE INCREMENTED EXP
U 1212, 0808,0038,0180,F800,0081,73AD           D_PACK.FP, J/EMODF.10           ; GET LEFTOVER FRACTION PART

                                       EMODF.10:
                                                ;-------------------------------;
                                                ALU_D, N_AMX.Z_TST,             ; SET COND CODES FROM FRACT
                                                WORD, INTRPT.STROBE,            ;
                                                STATE_K[ZERO],                  ; CLEAR STATE FOR RESULT STORE SECTION
U 13AD, 0001,543C,1980,F800,54B4,72A5           SC_K[ZERO], SC?                 ; SET FOR FLOAT, TEST FOR UNDERFLOW

                                       =0101    ;0101---------------------------;
                                                ALU_0(A),N&Z_ALU.V&C_0,D_0,     ; FLOATING UNDERFLOW - MAKE FRACT=0
U 12A5, 0F03,003D,0180,F800,0050,12AC           CALL[PSLFU]                     ;

                                       EMODFD:  ;0111---------------------------; ** EMODD ENTERS HERE **
                                                Q_ID[CES],                      ; NO UNDERFLOW
U 12A7, 0000,1A3C,31F0,2C00,0000,108C           PSL.V?,J/EMODF.11               ; SEE IF WE HAD AN INTEGER OVERFLOW

                                       =1101    ;1101---------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
U 12AD, 0F03,003C,01F8,F988,0000,108C           Q_0,D_0,RC[T1]_0,J/EMODF.11     ; CLEAR FRACT (INT=0 BY IMPLICATION)

                                                ;1111---------------------------; RETURN HERE IF PSL<fu>.er 1(SC.ne.0)
U 12AF, 0018,0038,F580,F9B8,0000,12B4           RC[T7]_K[.A],J/FLOAT.FAULT      ; T7 GETS UNDERFLOW TRAP CODE
```

```
                                          ;13055  108C:   ;ASSIGN THIS ADDRESS BECAUSE PCS CALLS IT
                                          ;13056  EMODF.11:
                                          ;13057          ;00*1100----------------;  ** EDIV ENTERS HERE **
                                          ;13058          SC_SC+1,                ;  SC SET TO 1/3 FOR FLOAT/DOUBLE
                                          ;13059          ID[T1]_D, D_RC[T1],     ;  SAVE FRACT<H>, GET FRACT<L>
                                          ;13060          CALL,INTERRUPT.REQ?,    ;
U 108C, 0810,0E39,C580,3D08,0080,C47E     ;13061          J/ASPC                  ;  EVALUATE INT.WL
                                          ;13062
                                          ;13063  108E:   ;ASSIGN THIS ADDRESS BECAUSE OF CONSTRAINT ON PREVIOUS INSTRUCTION
                                          ;13064          ;00*1110----------------;  INTEGER OVERFLOW OCCURRED
                                          ;13065          Q_D, D_Q.OR.K[.10],     ;  SET INTEGER OVERFLOW CODE IN C.E.S.
U 108E, 0819,2030,65E0,F800,0000,13B4     ;13066          J/EMODF.V               ;
                                          ;13067
                                          ;13068  10EC:   ;ASSIGN THIS ADDRESS BECAUSE OF CONSTRAINT ON PREVIOUS INSTRUCTION
                                          ;13069          ;11*1100----------------;  RETURN 60: INT.WL IS MEM MODE
                                          ;13070          RC[T2]_D,               ;  SAVE INT.WL ADDR
                                          ;13071          D_Q,                    ;  D GETS FRACT <L>
                                          ;13072          STATE_STATE+1,          ;  MARK FLAG FOR INT.WL R MODE
                                          ;13073          INTRPT.STROBE,
U 10EC, 0C01,003C,0180,F990,5400,D084     ;13074          J/EMODF.12              ;
                                          ;13075
                                          ;13076  10ED:   ;ASSIGN THIS ADDRESS BECAUSE OF CONSTRAINT ON PREVIOUS INSTRUCTION
                                          ;13077          ;11*1101----------------;  RETURN 61: INT.WL IS R MODE
                                          ;13078          D_Q,                    ;  D GETS FRACT <L>
                                          ;13079          RC[T2]_RLOG.RIGHT,      ;  SAVE REG # FOR INT.WL
                                          ;13080          INTRPT.STROBE,
U 10ED, 0C40,0038,0180,F990,5C00,1084     ;13081          J/EMODF.12              ;
                                          ;13082  =
                                          ;13083  EMODF.V:;----------------------;
                                          ;13084          ID[CES]_D, D_Q,         ;  WRITE CES WITH TRAP CODE, GO STORE RESULTS
U 13B4, 0C00,003C,3180,3C00,0000,108C     ;13085          J/EMODF.11              ;
```

```
                                          ;13086  =00****0
                                          ;13087  EMODF.12:;-----------------------;
                                          ;13088          ID[T2]_D, SC_SC+1,        ;  T2 SAVES FRACT <L>, SC GETS 2/4 (FLT/DBL)
                                          ;13089          CALL,INTERRUPT.REQ?,      ;
U 1084, 0000,0E3D,C980,3C00,0080,C47E     ;13090          J/ASPC                    ;  EVALUATE FRACT.WX MODE
                                          ;13091
                                          ;13092  =11****0;----------------------;  RETURN 60: FRACT.WX IS MEM MODE
                                          ;13093          RC[T4]_D,                 ;  SAVE FRACT.WX ADDR
                                          ;13094          STATE_STATE+K[.2],        ;  MARK FOR MEM MODE FOR FRACT
U 10E4, 0001,003C,0980,F9A0,1404,9114     ;13095          J/EMODF.14                ;  GOTO PROBE FRACT.WX
                                          ;13096
                                          ;13097          ;-----------------------;  RETURN 61: FRACT.WX IS R MODE
                                          ;13098          Q_ID[T0],                 ;  GET BACK INT.WL
U 10E5, 0000,173C,C1F0,2C00,0000,12EA     ;13099          STATE0?,J/EMODF.16        ;  INT.WL IS R OR M MODE?
                                          ;13100  =;END
                                          ;13101
                                          ;13102  =0
                                          ;13103  EMODF.14:;0----------------------;
                                          ;13104          VA_D,Q_D,                 ;  SETUP FRACTION ADDR FOR PROBE SUBROUTINE
                                          ;13105          D_D.RIGHT,                ;  GET D SHIFTED FOR PAGE BOUNDARY TEST
U 1114, 0601,003D,01E0,F800,0200,0D0D     ;13106          CALL,J/PRB.W              ;  GOTO PROBE FRACT.WX BEFORE WRITING ANY
                                          ;13107
                                          ;13108          ;1----------------------;  PROBE OK
                                          ;13109          Q_ID[T0],                 ;  GET BACK INT.WL
                                          ;13110          VA_RC[T4],                ;  GET BACK ADDR
U 1115, 0010,1738,C1F0,2D20,0200,12EA     ;13111          STATE0?,J/EMODF.16        ;  IS INT.WL MEM OR R MODE?
                                          ;13112  =;END
                                          ;13113  =10
                                          ;13114  EMODF.16:;0----------------------;   INT.WL IS R MODE
                                          ;13115          D_Q,Q_ID[T1],             ;  D GETS INT, Q GETS FRACT.WX
                                          ;13116          SC_RC[T2](EXP),           ;  GET BACK REG #
U 12EA, 0C10,0038,C5F0,2D10,0083,13B5     ;13117          J/EMODF.17                ;
                                          ;13118
                                          ;13119          ;1----------------------;  INT.WL IS MEM MODE
                                          ;13120          D_Q,Q_ID[T1],             ;  D GETS INT, Q GETS FRACT.WX
                                          ;13121          VA_RC[T2],                ;  GET ADDR
U 12EB, 0C10,0038,C5F0,2D10,0200,13B6     ;13122          J/EMODF.20                ;
                                          ;13123  =;END
                                          ;13124
                                          ;13125  EMODF.17:;-----------------------;
                                          ;13126          R(SC)_D,D_Q,              ;  WRITE INT.WL
U 13B5, 0C01,173C,0180,F8E8,0000,12F1     ;13127          STATE1?                   ;  IS FRACT MEM OR R MODE?
                                          ;13128
                                          ;13129  =01
                                          ;13130  EMODF.18:;0----------------------;   R MODE
                                          ;13131          R(PRN)_D,                 ;  WRITE FRACT.WF OR FRACT.WD <H>
                                          ;13132          Q_ID[T2],                 ;  GET BACK FRACT <L>
U 12F1, 0001,163C,C9F0,2CD8,0000,1008     ;13133          STATE7-4?,J/EMODF.19      ;  EMODF OR EMODD?
                                          ;13134
                                          ;13135          ;1----------------------;  MEM MODE
                                          ;13136          CACHE_D[INST.DEP],        ;  WRITE FRACT.WF OR FRACT.WD <H>
                                          ;13137          Q_RC[T1],                 ;  GET BACK FRACT <L>
U 12F3, 0010,D638,01C0,3108,0000,100C     ;13138          STATE7-4?,J/EMODF.22      ;  EMODF OR EMODD?
                                          ;13139  =;END
```

```
                                            ;13140  =1**0    ;0----------------------;  EMODF:
                                            ;13141  EMODF.19:CLR.IB.OPC,PC_PC+1,      ;  UPDATE IB, PC
U 1008, C000,003C,0180,F804,4000,0062       ;13142           J/IRD                    ;  GOTO NEXT INSTR
                                            ;13143
                                            ;13144           ;1----------------------;  EMODD:
                                            ;13145           R(PRN+1)_Q,              ;  WRITE FRACT.WD <L>
                                            ;13146           CLR.IB.OPC,PC_PC+1,      ;  UPDATE IB, PC
U 1009, C001,203C,0180,F8E4,4000,0062       ;13147           J/IRD                    ;  GOTO NEXT INSTR
                                            ;13148  =;END
                                            ;13149
                                            ;13150  EMODF.20:;----------------------;
U 13B6, 0000,003C,0180,3000,0000,13BC       ;13151           CACHE_D[LONG]            ;  STORE INT.WL
                                            ;13152
                                            ;13153           ;----------------------;
                                            ;13154           VA_RC[T4],D_Q,           ;  SET FRACT.WX ADDR
U 13BC, 0C10,1738,0180,F920,0200,12F1       ;13155           STATE1?,J/EMODF.18       ;  FRACT.WX MODE R OR MEM?
                                            ;13156
                                            ;13157  =1**0
                                            ;13158  EMODF.22:;0---------------------;   EMODF
                                            ;13159           CLR.IB.OPC,PC_PC+1,      ;  UPDATE IB, PC
U 100C, C000,003C,0180,F804,4000,0062       ;13160           J/IRD                    ;  GOTO NEXT INSTR
                                            ;13161
                                            ;13162           ;1----------------------;  EMODD
                                            ;13163           D_Q,                     ;  GET FRACT.WD <L>
                                            ;13164           VA_VA+4,                 ;  INC ADDR
U 100D, 0C00,003C,0180,F803,0000,03FD       ;13165           J/STOR.L                 ;
                                            ;13166  =;END
```

```
                                             ;13167  .TOC     ''       F & D floating point   : EMODD''
                                             ;13168
                                             ;13169  ;EMODD  (74)     MULR.RD, MULRX.RB, MULD.RD, INT.WL, FRACT.WD
                                             ;13170  ;ENTER WITH <RC 0, Q> = MULR.RD, D = MULRX.RB AT C.FORK.
                                             ;13171
                                             ;13172  :        COMPUTATIONAL METHOD - THE EXISTING CROSS-MULTIPLY TECHNIQUE,
                                             ;13173  :        WHICH WORKS FINE FOR MULD AND POLY, FALLS 1 BIT SHORT
                                             ;13174  :        IN ACCURACY FOR EMODD.  TO GET AROUND THIS, SPECIAL CODE IN THE
                                             ;13175  :        MULTIPLY DOUBLE SUBROUTINE (TRIGGERED BY A STATE BIT) DELETES
                                             ;13176  :        THE NORMALIZE BIT OF THE EXTENDED ARGUMENT, REDUCING ITS ACCURACY
                                             ;13177  :        TO 63 BITS.  AFTER THE MULTIPLY IS DONE, MORE SPECIAL CASE CODE
                                             ;13178  :        (IN EMODD THIS TIME) ADDS THE NON-EXTENDED ARGUMENT INTO THE
                                             ;13179  :        PRODUCT IN SUCH A WAY AS TO CONSTITUTE THE 64TH MULTIPLY STEP.
                                             ;13180  :        THIS SEEMS TO YIELD THE DESIRED ACCURACY.
                                             ;13181
                                             ;13182  386:
                                             ;13183  EMODD:   ;-------------------------------;
                                             ;13184           RC[T4]_D.OXT[BYTE],             ; GET 8 BIT EXT M'IER
                                             ;13185           D_Q,                            ; MOVE MULR <L> TO D
U 0386, 0C03,8E3D,0180,F9A0,0000,037E        ;13186           CALL,INTERRUPT.REQ?,J/SPEC      ; EVALUATE MULD.RD
                                             ;13187  396:
                                             ;13188           ;-------------------------------; RETURN WITH MULD IN <RC 2, D>
U 0396, 0000,003C,0180,F910,0000,1043        ;13189           LC_RC[T2],J/FL.ABS.1043         ;
                                             ;13190
                                             ;13191  1043:                                    ; ASSIGN THIS ADD BECAUSE PCS CALLS IT
                                             ;13192  FL.ABS.1043:
                                             ;13193           ;-------------------------------;
U 1043, 0000,003C,0180,F800,0000,1261        ;13194           J/EMODD.1                       ;  ** HACK TO AVOID CHANGING PROM **
                                             ;13195
                                             ;13196  =0**01   ;----------------------;
                                             ;13197  EMODD.1: RC[T1]_LC, STATE_K[.3],;  UNPACK AND MULTIPLY THE FRACTIONS
U 1261, 0010,0039,0D80,F988,1404,7003        ;13198            CALL[MULD.00]         ;  MULD HAS SPECIAL-CASE EXTENDER CODE
                                             ;13199
                                             ;13200  =1**01   ;0---------------------;  PRODUCT = 0
                                             ;13201           ID[T0]_D, STATE_FE,    ;  PREPARE TO SHARE CODE WITH EMODF
                                             ;13202           RC[T1]_0, N&Z_A[J.V&C_0,;  ZERO INTEGER AND FRACTION PARTS
U 1271, 0003,003C,C180,3D88,1450,73C1        ;13203                J/EMODD.6         ;  AND GO STORE IT ( MULD LEAVES 10 IN FE!)
                                             ;13204
                                             ;13205           ;1---------------------;  PRODUCT .NE. 0
                                             ;13206           RC[T2]_Q, STATE_K[.10], ;  SAVE LOW PRODUCT FRACTION, SET DBL FLAG
U 1273, 0001,203C,65F8,F990,1404,7041        ;13207           Q_0                    ;  CLEAR Q FOR CONSTRAINT HACKERY
                                             ;13208  =;END
```

```
                                      ;       ** FPA EMODD ENTERS HERE WITH MANTISSA IN <D,RC[T2]>, EXP IN SC,
                                      ;          STATE=10, SIGN IN SS, Q=0.

                                      =0****   ;-----------------------;
                                      EMODD.2: Q_RC[T2], SC_SC-K[.80],;   GO NORMALIZE FRACT AND GET INTEGER PART
                                                 FE_EALU,              ;   SAVE EXPONENT IN FE
U 1041, 0010,0D39,41C0,F910,0184,B2FA            D31?, CALL[EMODD.5]

                                               ;-----------------------;
                                               ID[T0]_D, D_RC[T2],     ;   RETURN FROM CVTFI - STORE INT FROM D,
U 1051, 0810,1A38,C1F8,3D10,1500,101C            Q_0, FE_STATE, PSL.V? ;   GET MANTISSA<L> IN D, TEST INT OVFLO

                                      =110*    ;0----------------------;   NO OVERFLOW - LEAVE STATE=10
                                               Q_RC[T1], CLK.UBCC,     ;   GET MANTISSA<H> IN Q & SET Z ON IT
                                                 ID[T2]_D, D_DAL.SC,   ;   DO 1ST PART OF 64-BIT FRACT SHIFT
U 101C, 0D10,1438,C9C0,3D08,0010,1025              SC?, J/EMODD.3      ;   WHILE WE CHECK IF ITS NECESSARY

                                               ;1----------------------;   OVERFLOW
                                               STATE_STATE.ANDNOT.FE,  ;   CLEAR STATE TO INDICATE OVFLO
                                                Q_RC[T1], CLK.UBCC,    ;   IF NUM=>1.0 WE WANT TO SHIFT THE INTEGER
                                                 ID[T2]_D, D_DAL.SC,   ;   PART OUT OF THE MANTISSA - START IT,
U 101E, 0D10,1438,C9C0,3D08,1410,5025                 SC?              ;   AND TEST IF WE REALLY OUGHTA DO IT.
                                               ;-----------------------;
                                      =0*101   ;BRANCH ON SC.GT.0 (NUM=>1.0) - ALSO CALLSITE FOR ADDD/PACKD
                                               ;0----------------------;   NUM < 1.0 (ALSO NUM=>1.0 CASE JOINS IN HERE)
                                      EMODD.3: D_Q, Q_RC[T2],          ;   GET UNSHIFTED FRACTION IN <D,Q>
                                                SC_SC+K[.80], FE_EALU, ;   PUT BIAS BACK INTO EXPONENT
U 1025, 0C10,1B39,41C0,F910,0184,867B             ALU?, CALL[ADDD.31]  ;   USE ADDD TO NORMALIZE, ROUND & PACK

                                               ;1----------------------;   NUM => 1.0
                                               RC[T2]_D, D_Q, Q_ID[T2],;   SAVE NEW FRACT<L>, SET UP TO SHIFT
U 1027, 0C01,003C,C9F0,2D90,0000,13BE                  J/EMODD.4       ;   FRACT<H> LEFT TO DESTROY INT PART

                                               ;-----------------------;   ADDD.31/PACKD RETURNS HERE - RESULT IN <D,RC1>
                                               STATE_K[.8], FE_K[.8],  ;   MONKEY BUSINESS TO SET STATE=10, SC=2
                                                 Q_K[.8].RIGHT2,       ;   SO WE CAN SHARE RESULT-STORE CODE WITH EMODF
U 1035, 0098,1638,01C0,F800,1504,7208              STATE4?             ;   MEANWHILE, TEST THE SAVED INT OVFLO FLAG
                                      =;END

                                      =***0    ;0----------------------;   INTEGER OVERFLOW OCCURRED
                                               SET.V, STATE_STATE+FE,  ;   SET STATE=10, SET V BIT (PACKD CLEARS IT)
U 1208, 0001,203C,0180,F800,14A2,92A7    SC_Q, J/EMODFD                ;   SET SC=2, SAVE PACKED RESULT<H>, JOIN EMODF

                                               ;1----------------------;   NO INTEGER OVERFLOW OCCURRED
                                                SC_Q, STATE_STATE+FE,  ;   SET SC=2, STATE=10 FOR EMODF
U 1209, 0001,203C,0180,F800,1482,92A7    J/EMODFD                      ;   GO STORE THE RESULTS
                                      =;END
                                               ;-----------------------;
U 13BE, 0D03,003C,0180,F800,0088,73C0 EMODD.4: D_DAL.SC, SC_0(A)       ;   MANTISSA SHIFT CONTINUED - SHIFT FRACT<H>

                                               ;-----------------------;
                                               Q_D, ALU_D, CLK.UBCC,   ;   SET Z ON HIGH LONGWD OF NEW FRACT
U 13C0, 0001,003C,01E0,F800,0010,1025                  J/EMODD.3       ;   AND GO NORMALIZE, ROUND AND PACK IT
```

```
                                        ;13262  ;       INTERFACE BETWEEN EMODD AND CVTFI - CALLED AS MICROSUBROUTINE
                                        ;13263
                                        ;13264          ;----------------------;
                                        ;13265  =*10    ;BRANCH ON D31 (PRODUCT IS NORMALIZED) (Q = 0)
                                        ;13266          ;0---------------------;  PRODUCT NOT NORMALIZED
                                        ;13267  EMODD.5: ALU_0+K[.1], Q_Q.LEFT, ;  SHIFT THE FRACTION LEFT A BIT
                                        ;13268           D_D.LEFT, SI/DIVD,     ;
U 12FA, 051B,0014,0428,F800,0184,B2FB   ;13269           SC_SC-K[.1], FE_EALU  ;  ADJUST EXPONENT TO MATCH
                                        ;13270
                                        ;13271          ;1---------------------;  PRODUCT NORMALIZED
U 12FB, 0003,003C,0180,F988,0000,0946   ;13272          RC[T1]_0, J/CVTFI.1    ;  INITIALIZE MANTISSA<H> HOLDER & CALL CONVERT
                                        ;13273  =;END
                                        ;13274
                                        ;13275
                                        ;13276
                                        ;13277          ;----------------------;  CONTINUATION OF EMODD PROD=0 CASE
                                        ;13278  EMODD.6: SGN/CLR.SD+SS,        ;  CLEAR SIGNS (MAY NOT BE BECESSARY)
U 13C1, 0000,003C,0987,F800,0084,708C   ;13279           SC_K[.2], J/EMODF.11  ;  SET D.P. FLAG IN SC AND JOIN EMODF
```

```
                                       .TOC    ''        F & D floating point   : POLYF''

                                       ;          POLYF   (55)     ARG.RF, DEGREE.RW, TBLADDR.AB
                                       ;ENTER AT C.FORK WITH Q HAS ARG.RX, D HAS DEGREE.RW.
                                       ;WHEN DONE, R0 = RESULT, R2 = 0, R1 = 0, R3 = TABLE ADDR + DEG*4 + 4.
                                       ;IN PROCESSING, R0 = PARTIAL RESULT, R2 = DEG, R1 = ARG, R3 = NEXT TABLE ADDR.

                                       3C2:
                                       POLYF:   ;---------------------------------;
U 03C2, 0803,403C,0180,F800,0000,1012           D_D.OXT[WORD],J/POLYF.0            ; GET DEGREE.RW
                                       0C2:
                                       POLYF.FPD:
                                                ;---------------------------------;
                                                D_R[R2],Q_0,SC_K[.FFF8],           ; SETUP TO GET PC DELTA
U 00C2, 0800,003C,71F8,FA10,0084,701D           J/FL.ABS.101D                      ;

                                       101D:                                       ;ASSIGN THIS ADD BECAUSE PCS CALLS IT
                                       FL.ABS.101D:
                                                ;---------------------------------;
U 101D, 0D00,003C,0180,F800,0000,1310           D_DAL.SC                           ;  PC DELTA IN D[07:00]

                                       =C0      ;00--------------------;
                                                PC&VA_D.OXT[BYTE]+PC,  ;   BYPASS SPECIFIERS
U 1310, 0017,8015,0180,F801,0200,0E16           CALL, J/SETFPD         ;   NEED TO RE-SETUP ID[FPDA]

                                       =10      ;10--------------------;   READ ERRORS COME HERE
U 1312, 0014,0038,01C0,F800,0000,0EB8           Q_PC, J/BAKUP.PC       ;   BACKUP PC AND CAUSE A TRAP

                                                ;11--------------------;   RETURN FROM SETFPD HERE
U 1313, 0000,003C,0180,F800,4000,1082           INTRPT.STROBE, J/POLYF.2;  WASTE - HOWEVER TOO TOUGH TO SHARE SETFPD CALL
                                       =;END

                                       12C2:
                                       POLY.C:  ;----------------------;   SET COND CODES AFTER DONE FOR BOTH POLYF/D
                                                EALU_SC,ALU_R[R0],     ;   SC SHOULD HAVE THE EXP ID
U 12C2, 0000,C03C,0180,FA00,0070,13C3           SET.CC(INST)           ;   SET COND CODES

                                       POLY.0:  ;----------------------;   FLUSH IB
                                                ALU_PC,FLUSH.IB,       ;   FLUSH IB IN CASE RE-ENTERED
U 13C3, 2014,0038,0180,F800,4200,00AB           J/IB.FILL              ;

                                       12C3:    ;----------------------;   SET COND CODES AFTER DONE FOR BOTH POLYF/D
                                                EALU_SC,ALU_R[R0],     ;   SC SHOULD HAVE THE EXP ID
                                                SET.CC(INST),          ;   SET COND CODES
U 12C3, 0000,C03C,0180,FA00,0070,13C3           J/POLY.0               ;   GOTO FLUSH IB

                                       448:
                                       POLY.CC:
                                                ;---------------------------------; SET CC AFTER DONE FOR BOTH POLYF/D
                                                EALU_SC,ALU_R[R0],                 ; SC SHOULD HAVE THE EXP ID
                                                SET.CC(INST),                      ; SET COND CODES
U 0448, C000,C03C,0180,FA04,4070,0062           CLR.IB.OPC,PC_PC+1,J/IRD           ; UPDATE IB, PC
```

```
                                     1012:    ;ASSIGN THIS ADDRESS BECAUSE PCS CALLS IT
                                     POLYF.0:;----------------------;
                                              ALU_D.ANDNOT.K[.1F],   ;  CHECK IF DEGREE > 31
                                              CLK.UBCC,              ;  CLOCK IN ALU.Z
U 1012, 0C19,0024,8DE0,F800,0010,13C4         D_Q,Q_D                ;  D GETS ARG, Q GETS DEGREE
                                     =;END

                                              ;----------------------;
                                              ID[T0]_D,              ;  T0 SAVES ARG
                                              ALU_Q(B), RC[T2]_ALU,  ;  RC 2 SAVES DEGREE, SET ALU.Z ON DEGREE
                                              FE_D(EXP),CLK.UBCC,    ;  FE GETS ARG EXP, CLOCK IN FOR ZERO CHECK
                                              INTRPT.STROBE,         ;  ENABLE INTERRUPTS
U 13C4, 001D,0138,C180,3D90,4118,70A2         Z?                     ;  CHECK RANGE OF DEGREE IF LEGAL?

                                     =01****0;----------------------;  NO: OUT OF RANGE
U 10A2, 0000,003C,0180,F800,0000,0106         J/RSVOPR               ;  DEGREE > 31: ILLEGAL

                                     =01****1;----------------------;  YES: IN RANGE
                                              ALU_D,CHK.FLT.OPR,     ;  CHECK IF ARG IS -0
                                              CALL,INTERRUPT.REQ?,   ;
U 10A3, 0001,0E3D,0180,F800,0800,047E         J/ASPC                 ;  GET COEF

                                     =11****1;----------------------;  ASPC ALWAYS RETURNS 60 IN THIS CASE
                                              Q_D,                   ;  TABLE ADDR
                                              D[LONG]_CACHE,         ;  GET COEF C0
                                              R[R15]_ALU.RIGHT,      ;
U 10E3, 0058,0038,45E0,42F8,0000,13C5         SI/ZERO,ALU_K[.8000]   ;  SETUP FOR 4080 (FP 1.)
                                     =;END

                                              ;----------------------;
U 13C5, 0001,003C,0180,F800,0800,13C9         ALU_D,CHK.FLT.OPR      ;  CHECK 1ST COEF C0 FOR -0

                                              ;----------------------;
U 13C9, 0001,203C,C1F0,2E98,0000,1224         R[R3]_Q,Q_ID[T0]       ;  R3 STORES TABLE ADDR, Q GETS ARG

                                     =0       ;----------------------;
                                              ID[T1]_D,              ;  T1 SAVES C0
                                              ALU_Q,                 ;  R1 GETS ARG
                                              LC_RC[T2]&R1_ALU,      ;  LATCH DEGREE
U 1224, 0001,203D,C580,3F90,0000,13E8         CALL,J/POLY.PC         ;  GO GET PC DELTA

                                              ;----------------------;
                                              R[R2]_D.OR.LC,         ;  R2 GETS PC DELTA, DEGREE
U 1225, 0011,0030,C5F0,2E90,0000,1328         Q_ID[T1]               ;  Q GETS C0
                                     =;END

                                     =00      ;00--------------------;
                                              SC_Q(EXP),             ;  SC GETS C0 EXP
                                              D_LC,                  ;  D GETS DEGREE
U 1328, 0811,2039,0180,F800,0088,6E16         CALL,J/SETFPD          ;  SET FPD

                                     =10      ;10--------------------;  READ ERROR, POLYF FPD PACKING ROUTINE
U 132A, 0014,0038,01C0,F800,0000,0EB8         Q_PC, J/BAKUP.PC       ;  BACK UP PC AND CAUSE EXCEPTION

                                              ;11--------------------;
U 132B, 0018,0038,41C0,FA78,0000,13CA         Q_K[.80],LAB_R[R15]    ;  GET 4080 (FP 1.)
```

```
                                                    ;-----------------------;
                                                    Q_ID[.1],               :   GET BACK 1ST COEF C0
U 13CA, 001C,0014,C5F0,2E80,0000,13CB               R[R0]_LA+Q              :   RUNNING SUM IS 1.0 BEFORE C0 TESTED FOR -0

                                                    ;-----------------------;
U 13CB, 0000,003C,0180,FA18,0000,13CD               LAB_R[R3]               :   LATCH TABLE ADDR

                                                    ;-----------------------;
                                                    R[R3]&VA_LA+K[.4],      :   INC TABLE ADDR
                                                    SGN/CLR.SD+SS,          :   EASE UP ON EALUZ CONSTRAINT
U 13CD, 0018,0D14,1187,FA98,0200,1064               D.NE.0?                 :   IS DEGREE 0?

                                           =10*     ;0----------------------;   DEGREE = 0
                                                    CLR.FPD,                :   CLR PSL <FPD> BIT
                                                    R[R0]_Q,                :   R0 GET COEF C0
U 1064, 0001,343C,0180,FA80,2000,1301               SC.GT.0?,J/POLYF.5      :   IS C0 0?

                                                    ;1----------------------;   DEGREE .NE. 0
                                                    R[R0]_Q, INTRPT.STROBE, :   R0 GETS COEF C0
U 1066, 0001,323C,0180,FA80,4000,1082               EALU.Z?                 :   IS ARGUMENT 0?
                                           =;END

                                           =*01*
                                           POLYF.2:;0-------------------;   NO:
                                                    Q_R[R1](FRAC),          :   ARG FRAC
                                                    SC_R[R1](EXP),          :   ARG EXP
                                                    SS_ALU15,               :   ARG SIGN
                                                    INTERRUPT.REQ?,         :
U 1082, 0000,0E3C,01C9,FA08,0083,1336               J/POLYF.8               :

                                                    ;1----------------------;   YES:  ARGUMENT = 0
                                                    R[R0]_K[ZERO],          :   ZERO OUT THE PARTIAL PRODUCT
U 1086, 0118,0038,1980,FA80,0000,13D0               D_D.LEFT2               :   D GETS DEGREE*4
                                           =;END
                                                    ;-----------------------;
U 13D0, 001C,2014,0180,FA98,0000,13D1               R[R3]_LA+D              :   ADVANCE THE EXECUTION OF THE POLYNOMIAL
                                                                            :   TO THE LAST COEFFICIENT
                                                    ;-----------------------;
                                                    R[R2]_K[.1], DT/BYTE,   :   BY BUMPING THE TABLE ADDR AND DEGREE
U 13D1, 0018,8038,0580,FA90,0000,1082               J/POLYF.2               :   NOW JOIN THE ORDINARY ITERATION CODE

                                           =*01
                                           POLYF.5:;0-------------------;   C0 IS 0
U 1301, 0018,0038,1980,FA80,0000,1303               R[R0]_K[ZERO]           :   SET RESULT 0

                                                    ;1----------------------;   C0 IS NOT 0
U 1303, 0003,003C,0180,FA90,0000,13E6               R[R2]_0, J/POLYF.22     :   CLR R2 AND GO EXIT
                                           =;END
```

```
                                         ;13435  =110
                                         ;13436  POLYF.8:;0---------------------;  NO INTERRUPT REQ
                                         ;13437          D_R[R0](FRAC),         ;  GET PARTIAL RESULT FRAC
                                         ;13438          FE_R[R0](EXP),         ;  PARTIAL RESULT EXP
                                         ;13439          SS_SS.XOR.ALU15&SD_ALU15,;  SS PARTIAL RESULT SIGN
                                         ;13440          CLR.UBCC,              ;  CLOCK IN IF ZERO (EALU.Z)
U 1336, 0900,003C,0185,FA00,0118,73D2    ;13441          J/POLYF.9              ;
                                         ;13442
                                         ;13443          ;1---------------------;  YES: INTERRUPT REQ
U 1337, 0000,003C,0180,F800,0000,121C    ;13444          J/POLY.INT             ;  INTRUPT ENTRY TO CLR TP AND BACKUP PC,R'S
                                         ;13445  =;END
                                         ;13446  POLYF.9:;----------------------;
                                         ;13447          FE_SC+FE,SD_SS,        ;  ADD EXP'S, SD GETS PART PROD SIGN
                                         ;13448          RC[T0]_Q,Q_0,          ;
U 13D2, 0001,323C,01FC,F980,0100,9343    ;13449          EALU.Z?                ;  PARTIAL RESULT = 0?
                                         ;13450
                                         ;13451  =*011    ;0---------------------;  NO:
                                         ;13452          LC_RC[T0],             ;  LATCH M'CAND * 2 (ARG)
                                         ;13453          SC_K[.FFF9],           ;  SETUP DAL SHF COUNT
U 1343, 0000,003C,BD80,F900,0084,73DC    ;13454          J/POLYF.10             ;
                                         ;13455
                                         ;13456          ;1---------------------;  YES:RETURN NEXT COEF AS PARTIAL PROD
                                         ;13457          SC_K[.7F],D_0,         ;  PARTIAL RESULT EXP SET TO 0
                                         ;13458          SGN/CLR.SD+SS,         ;  FRAC SET TO 0, SIGN SET TO 0
U 1347, 0F00,003C,5987,FA18,0284,73D3    ;13459          VA_R[R3]               ;  LATCH TABLE ADDR
                                         ;13460  =;END
                                         ;13461  ;
                                         ;13462  ;NOTE THAT THE NORMAL ADD ROUTINE IS SKIPPED IF THE PARTIAL PRODUCT
                                         ;13463  ;IS ZERO BECAUSE THE ADD ROUTINE DOESN'T ALWAYS HANDLE THE ZERO CASE
                                         ;13464  ;CORRECTLY
                                         ;13465  ;
                                         ;13466          ;----------------------;
U 13D3, 0000,003C,0180,4000,0000,13D9    ;13467          D[LONG]_CACHE          ;  GET NEXT COEFFICIENT
                                         ;13468
                                         ;13469          ;----------------------;
                                         ;13470          R[R0]_D,D_D(FRAC),     ;  SAVE COEF,UNPACK IT AND TEST FOR RES OPR
                                         ;13471          SC_D(EXP),SS_ALU15,    ;
U 13D9, 0901,003C,0181,FA80,0893,13DB    ;13472          CLR.UBCC,CHK.FLT.OPR   ;
                                         ;13473
                                         ;13474          ;----------------------;
                                         ;13475          R[R3]&VA_LA+K[.4],     ;  INCREMENT TABLE ADDRESS
U 13DB, 0018,0C14,1180,FA98,0200,12BB    ;13476          SC.NE.0?               ;  IS COEF ZERO?
                                         ;13477
                                         ;13478  =1011    ;1011------------------;  SC=0
U 12BB, 0018,0038,1980,FA80,0000,12BF    ;13479          R[R0]_K[ZERO]          ;  MAKE SURE ZERO IS CLEAN
                                         ;13480
                                         ;13481          ;1111------------------;  SC NE ZERO
                                         ;13482          LAB_R[R2],FE_K[.1F],   ;  LATCH DEGREE,SET UP FE
U 12BF, 0000,003C,8D80,FA10,0104,73E3    ;13483          J/POLYF.17             ;  REJOIN CODE AFTER ADD
```

```
                                        ;13484
                                        ;13485   POLYF.10:;-----------------------;
                                        ;13486             D_DAL.SC,                ;   SHF M'IER IN POSITION
                                        ;13487             R[R15]_LC.RIGHT,SI/ZERO,;   R15 GET M'CAND (ARG)
U 13DC, 0D50,0038,8580,FAF8,0084,7214   ;13488             SC_K[.C]                 ;   LOOP CT SET FOR 13. LOOPS
                                        ;13489
                                        ;13490   =0*       ;0-----------------------;
                                        ;13491             D_D.RIGHT2,SI/ZERO,      ;   BEGIN SHF'G M'CAND
                                        ;13492             ALU_0(A),                ;   LATCH ALU[01:00]
                                        ;13493             LAB_R[R15],              ;   LATCH M'CAND
U 1214, 0203,0C3D,0180,FA78,0000,0350   ;13494             CALL,MUL?,J/MULPP        ;   CALL MUL SUBRTN - PROD HAS 4-5 LEADING 0'S
                                        ;13495
                                        ;13496             ;1-----------------------;   RETURN FROM MUL SUBRT
                                        ;13497             SC_K[.3],                ;   GET READY TO ALIGN RESULT FRACTION
U 1216, 0C00,003C,0DE0,F800,0084,73DD   ;13498             D_G,Q_D                  ;   D GETS HIGH PROD, Q GETS LOW PROD
                                        ;13499   =;END
                                        ;13500             ;------------------------;
U 13DD, 0D00,003C,01F8,F800,0081,13E0   ;13501             SC_FE, D_DAL.SC, Q_0     ;   FRACTION NOW HAS 31 OR 30 SIGNIFICANT BITS
                                        ;13502
                                        ;13503             ;------------------------;
                                        ;13504             SC_SC-SHF.VAL,           ;   NORMALIZE
                                        ;13505             D_DAL.NORM,              ;
U 13E0, 0E00,003C,0180,FA18,028C,B3E1   ;13506             VA_R[R3]                 ;   LATCH DEGREE COUNT
```

```
                                    ;13507  POLYF.12:
                                    ;13508          ;---------------------------------;
                                    ;13509          RC[T6]_ALU,ALU_PACK.FP,           ; SAVE PART PROD SIGN
                                    ;13510          FE_SC-K[.7F],CLK.UBCC,            ; ADJUST RESULT EXP
                                    ;13511          Q_D,                              ; Q GETS MUL RESULT
U 13E1, 0008,0038,59E0,41B0,0114,B3E2 ;13512       D[LONG]_CACHE                     ; GET NEXT COEF
                                    ;13513
                                    ;13514          ;---------------------------------;
                                    ;13515          Q_Q.RIGHT,SI/ZERO,                ; PUT PRODUCT FRACTION IN ADDD FORMAT
                                    ;13516          R[R15]_D,D_D(FRAC),               ; COEF FRAC
U 13E2, 0901,003C,01B1,FAF8,0883,10C6 ;13517       SC_D(EXP),SS_ALU15,CHK.FLT.OPR    ; COEF EXP, COEF SIGN, CHECK FOR -0
                                    ;13518
                                    ;13519  =0*1**0110
                                    ;13520          ;0*1**0110------------------------;
                                    ;13521          D_Q,Q_D,                          ; D GETS DST FRAC, Q SRC FRAC
                                    ;13522          R[R3]&VA_LA+K[.4],                ; UPDATE TABLE ADDR
                                    ;13523          SC_SC-FE,CLK.UBCC,                ; GET EXP DIFF, CLOCK IN FOR EXP'S DIFF
U 10C6, 0C18,1415,11E0,FA98,0290,B321 ;13524       SC.GT.0?, CALL[POLYF.14]          ; IS  COEF 0?
                                    ;13525
                                    ;13526  =1*1**0110
                                    ;13527          ;1*1**0110------------------------; RESULT 0
U 11C6, 0000,003C,0180,FA10,0000,12DD ;13528       LAB_R[R2],J/POLYF.19              ; LATCH DEGREE
                                    ;13529
                                    ;13530  =1*1**1110
                                    ;13531  POLYF.13:
                                    ;13532          ;1*1**1110------------------------; RESULT NON-0, NO CARRY FROM FRAC
                                    ;13533          LAB_R[R2],                        ; LATCH DEGREE
                                    ;13534          FE_K[.1F],                        ; SETUP TO DISREGARD UNDERFLOW FLAG
U 11CE, 0000,143C,8D80,FA10,0104,72D1 ;13535       SC?,J/POLYF.26                    ; CHECK FOR OVER/UNDER FLOW, 0, ETC.
                                    ;13536
                                    ;13537          ;1*1**111-------------------------; RESULT NON-0, CARRY FROM FRAC
                                    ;13538          D_D.RIGHT,SC_SC+1,                ; ROUND UP BY 1
U 11CF, 0600,003C,0180,F800,0080,D1CE ;13539       SI/ZERO,J/POLYF.13                ;
                                    ;13540  =*01
                                    ;13541  POLYF.14:
                                    ;13542          ;*01------------------------------; COEFFICIENT IS 0
                                    ;13543          SC_FE, ALU_D, CLK.UBCC,           ; SUM = PARTIAL PRODUCT, ROUNDED.
U 1321, 0001,003C,0180,F800,0091,0604 ;13544       J/NEGCK                           ; USE ADDF CODE TO CHECK FOR 0 AND ROUND
                                    ;13545
                                    ;13546          ;*11------------------------------; COEFFICIENT .NE. 0
                                    ;13547          ID[T1]_D,ALU_RC[T6],              ; SAVE DST OPR (PART PROD),
                                    ;13548          SS_SS.XOR.ALU15&SD_ALU15,         ; FIX SIGN INDICATORS FOR FADD
U 1323, 0010,1438,C585,3D30,0000,1355 ;13549       SC.GT.0?                          ; SEE IF WE SHOULD NEGATE EXPONENT DIFF
                                    ;13550
                                    ;13551  =101    ;101------------------------------; EXP DIFF <= 0 - NO NEED TO NEGATE
U 1355, 0000,123C,0180,F800,0000,05A2 ;13552       EALU?,J/ADDFSH                    ; BREAK OUT FADD CASES AND ADD
                                    ;13553
                                    ;13554          ;111------------------------------; EXP DIFF > 0 - MUST NEGATE
                                    ;13555          ALU_0-K[SC],SC_ALU,               ; NEGATE SC THRU MAIN DATA PATH
U 1357, 001B,1200,1D80,F800,0082,05A2 ;13556       EALU?,J/ADDFSH                    ; NOW GO DO THE FADD.
                                    ;13557
                                    ;13558  ;YOU MAY ASK, WHY GO THROUGH THIS RIGAMAROLE WHEN WE HAVE AN EALU FUNCTION TO
                                    ;13559  ;TAKE THE NEGATIVE ABSVAL OF (SC-FE)? THE ANSWER IS THAT SINCE THE MULTIPLY IS
                                    ;13560  ;ALLOWED TO UNDERFLOW, THE RANGE OF THE EXPONENT DIFFERENCE IS LARGER THAN THE
                                    ;13561  ;SIZE OF THE NABS ROM.
```

```
                                          ;          COME HERE AFTER ADD WITH FRAC IN D, EXP IN SC, BEN/SC
                                          =0001
                                          POLYF.26:
                                                     ;0001---------------------------; SC.eq.0, UNDERFLOW
U 12D1, 0000,003D,3DF0,2C00,0000,12AE                Q_ID[PSL],CALL[PSLFU.A]         ; Get PSL<fu> & go see if it is set

                                                     ;0011---------------------------; SC.eq.[01 to FF], NO OVER/UNDERFLOW
                                                     EALU_SC,R[R0]_PACK.FP,          ; R0 STORES RESULT
U 12D3, 0008,0038,0180,FA80,0100,13E3                FE_EALU,J/POLYF.17              ; FE GETS RESULT EXP TO SET CC LATER

                                                     ;0101---------------------------; SC.lss.0, UNDERFLOW
U 12D5, 0000,003D,3DF0,2C00,0000,12AE                Q_ID[PSL],CALL[PSLFU.A]         ; Get PSL<fu> & go see if it is set

                                                     ;0111---------------------------; SC.gt.0, OVERFLOW
                                                     D_K[.8],                        ; D GETS OVERFLOW TRAP CODE
U 12D7, 0818,0038,0180,FA18,0000,13F0                LAB_R[R3],J/POLYF.FAULT         ; FETCH TABLE ADDRESS
                                          =1101
                                          POLYF.19:
                                                     ;1101---------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
U 12DD, 0003,003C,1980,FA80,0104,73E3                R[R0]_0,FE_K[ZERO],J/POLYF.17   ; PRETEND UNDERFLOW DIDN'T HAPPEN

                                                     ;1111---------------------------; RETURN HERE IF PSL<fu>.eq.1(SC.ne.0)
                                                     D_K[.A],                        ; D GETS UNDERFLOW TRAP CODE
U 12DF, 0818,0038,F580,FA18,0000,13F0                LAB_R[R3],J/POLYF.FAULT         ; FETCH TABLE ADDRESS

                                          POLYF.17:
                                                     ;-------------------------------;
                                                     R[R2]_LA-K[.1],BYTE,            ; DECREMENT DEGREE COUNT IN R2<7:0>
U 13E3, 0018,8000,0580,FA90,0010,13E4                CLK.UBCC                        ; AND SET Z IF WE ARE DONE

                                                     ;-------------------------------;
U 13E4, 0000,013C,0180,F800,4000,1234                INTRPT.STROBE,Z?                ; ENABLE INTERRUPTS - IS COUNT UP?

                                          =0         ;0------------------------------; NO:
                                                     Q_R[R1](FRAC),                  ; ARG FRAC
                                                     SC_R[R1](EXP),                  ; ARG EXP
                                                     SS_ALU15,                       ; ARG SIGN
                                                     INTERRUPT.REQ?,                 ; ANY INTERRUPT REQUEST?
U 1234, 0000,0E3C,01C9,FA08,0083,1336                J/POLYF.8                       ; GOTO NEXT LOOP

                                                     ;1------------------------------; YES: THIS IS THE END
U 1235, 0018,0038,1980,FA90,2000,13E6                CLR.FPD,R[R2]_K[ZERO]           ; CLEAR PSL<FPD>, CLEAR R2

                                          POLYF.22:
                                                     ;-------------------------------;
                                                     R[R1]_K[ZERO],                  ; CLR R1
U 13E6, 0018,003B,1980,FA88,0000,12C2                SUB/SPEC,J/POLY.C               ; GOTO SET COND CODES
```

```
                                          ; ROUTINE TO GET PC DELTA FOR POLY
                                          POLY.PC:
                                                   ;---------------------------------; Q GETS PC
U 13E8, 0017,0018,01C0,F800,0000,13E9              Q_0+PC.RLOG                        ; PUSH RLOG SO PCSV REF WON'T KILL IT

                                                   ;---------------------------------;
U 13E9, 0801,2000,0180,F800,1C00,13EC              D_Q-PCSV                           ; GET PC DELTA

                                                   ;---------------------------------;
U 13EC, 0803,803C,01F8,F800,0084,73ED              D_D.0XT[BYTE],SC_K[.8],Q_0         ; 0EXT PC DELTA, SETUP TO SHIFT PC DELTA

                                                   ;---------------------------------;
U 13ED, 0D00,003E,C1F0,2C00,0000,0001              D_DAL.SC,Q_ID[T0],RETURN1          ; SHIFT PC DELTA, GET DEGREE FOR POLYD


                                          =0*
                                          POLY.INT:
                                                   ;0*-------------------------------; PUT PC WHERE BAKUP.PC WANTS IT AND
U 121C, 0014,0039,01C0,F800,0000,0EB8              Q_PC,CALL[BAKUP.PC]                ; GO DO IT

                                                   ;1*-------------------------------;
U 121E, 0000,003C,0180,F800,0000,04FA              J/INT.I                            ; GO ALLOW INTERRUPT


                                          POLYF.FAULT:
                                                   ;---------------------------------; Enter here with fault code in D
U 13F0, 0018,0000,1180,FA98,0000,13F2              R[R3]_LA-K[.4],J/POLY.FAULT        ; DE-INCREMENT TABLE ADDRESS

                                          POLYD.FAULT:
                                                   ;---------------------------------; Enter here with fault code in D
U 13F1, 0018,0000,0180,FA98,0000,13F2              R[R3]_LA-K[.8],J/POLY.FAULT        ; DE-INCREMENT TABLE ADDRESS

                                          POLY.FAULT:
                                                   ;---------------------------------;
U 13F2, 0001,003C,0180,F9B8,0000,13F3              RC[T7]_D                           ; PUT FAULT CODE WHERE IT WILL BE SAFE

                                                   ;---------------------------------;
                                                   RC[T0]_K[.34],                     ; TO VECTOR ID OF FLOATING FAULTS
U 13F3, 001B,0038,2980,F980,1408,7330              STATE_0(A)                         ; CLEAR STATE(WILL BE SET TO 1 LATER)

                                          =00      ;00-------------------------------;
                                                   Q_PC,RC[PC.SV]_PC,                 ; SET UP PC WHERE 'BAKUP.PC'' WANTS IT
U 1330, 0014,0039,01C0,F9E0,0000,0EB8              CALL[BAKUP.PC]                     ; GO BACK UP PC

                                          =10      ;10-------------------------------; RETURN HERE FROM 'BAKUP.PC''
                                                   Q_ID[PSL],                         ; GET PSL INTO Q
                                                   STATE_STATE+1,                     ; STATE=1 TO INDICATE PARAMETERS
U 1332, 0000,003D,3DF0,2C00,1400,CDE8              CALL[EXCPT1]                       ; GO INITIALIZE FAULT

                                                   ;11-------------------------------; RETURN HERE FROM 'EXCPT1''
U 1333, 0810,0038,0180,F938,0000,13F4              D_RC[T7]                           ; GET FAULT CODE

                                          POLY.FLOAT.FAULT.A:
                                                   ;---------------------------------;
U 13F4, 0018,0000,1180,FAF0,0200,002A              R[SP]&VA_LA-K[.4],J/EXCPT2         ; DEC STACK POINTER & GO PSH FLT CODE
```

```
                                           ;13664  .TOC    "       F & D floating point   : POLYD"
                                           ;13665
                                           ;13666  ;       POLYD   (75)    ARG.RD, DEGREE.RW, TBLADDR.AB
                                           ;13667  ;ENTER AT C.FORK WITH <RC 0, Q> HAVE ARG.RX, D HAS DEGREE.RW.
                                           ;13668  ;WHEN DONE, <R0, R1> = RESULT, R2 = 0, R3 = TABLE ADDR + DEG*8 + 8,
                                           ;13669  ;       R4 = 0, R5 = 0.
                                           ;13670  ;IN PROCESSING, <R0, R1> = PARTIAL RESULT, R2 = DEG, R3 = NEXT TABLE ADDR,
                                           ;13671  ;       <R4, R5> = ARG.
                                           ;13672
                                           ;13673  385:
                                           ;13674  POLYD:  ;-------------------------------;
                                           ;13675          D_D.OXT[WORD],                  ; GET DEGREE.RW
U 0385, 0803,403C,0580,F800,1404,7013      ;13676          STATE_K[.1],J/POLYD.0           ; SET POLYD FLAG
                                           ;13677  0C3:
                                           ;13678  POLYD.FPD:
                                           ;13679          ;-------------------------------;
                                           ;13680          D_R[R2],Q_0,SC_K[.FFF8],        ; SETUP TO GET PC DELTA
U 00C3, 0800,003C,71F8,FA10,0084,7014      ;13681          J7FL.ABS.1014                   ;
                                           ;13682
                                           ;13683  1014:                                   ;ASSIGN THIS ADD BECAUSE PCS CALLS IT
                                           ;13684  FL.ABS.1014:
                                           ;13685          ;-------------------------------;
U 1014, 0D00,003C,0180,F800,0000,1338      ;13686          D_DAL.SC                        ;  PC DELTA IN D[07:00]
                                           ;13687
                                           ;13688  =00     ;00----------- --------;  
                                           ;13689          PC&VA_D.OXT[BYTE]+PC,  ;  BYPASS SPECIFIERS
U 1338, 0017,8015,0180,F801,0200,0E16      ;13690          CALL, J/SETFPD         ;  MUST ESTABLISH A VALID FPD ERROR HANDLER!
                                           ;13691
                                           ;13692  =10     ;10--------------------;  MEM MGMT CODE COMES HERE ON READ ERRORS
U 133A, 0014,0038,01C0,F800,0000,0EB8      ;13693          Q_PC, J/BAKUP.PC       ;  BACK UP PC AND CAUSE EXCEPTION
                                           ;13694
                                           ;13695          ;11--------------------;  SETFPD RETURNS HERE
U 133B, 0000,003C,0180,FA18,0200,1366      ;13696          VA_R[R3], J/POLYD.9    ;  GET TABLE ADDR, JUMP BACK INTO LOOP
                                           ;13697  =;END
```

```
                                        1013:     ;ASSIGN THIS ADDRESS BECAUSE PCS CALLS IT
                                        POLYD.0:;-----------------------;
                                                  LC_RC[T0],            ;   LATCH ARG <H>
                                                  ALU_D.ANDNOT.K[.1F],  ;   CHECK IF DEGREE > 31
U 1013, 0019,0024,8D80,F900,0010,13F6             CLK.UBCC              ;   CLOCK IN ALU.Z

                                                  ;---------------------;
                                                  ID[T0]_D,             ;   SAVE DEGREE
                                                  D_LC,CHK.FLT.OPR,     ;   CHECK ARG.RX FOR -0
                                                  SC_LC(EXP),           ;   ARG EXP
U 13F6, 0810,0138,C180,3C00,4883,10B8             INTRPT.STROBE, Z?     ;   DEGREE > 31?

                                        =01****0;0----------------------;   DEGREE > 31
U 10B8, 0000,003C,0180,F800,0000,0106             J/RSVOPR              ;   RESERVED OPERAND

                                                  ;1--------------------;   DEGREE IN RANGE OF 0 TO 31.
                                                  ID[T2]_D,FE_SC,       ;   T2 SAVES ARG <H>, FE SAVES ARG EXP
                                                  RC[T2]_Q,             ;   RC 2 SAVES ARG <L>
                                                  CALL,INTERRUPT.REQ?,  ;
U 10B9, 0001,2E3D,C980,3D90,0100,C47E             J/ASPC                ;   GO GET TABLE ADDR

                                        =11****1;-----------------------;   ASPC ALWAYS RETURNS 60 IN THIS CASE
                                                  R[R15]_D,             ;   SAVE TABLE ADDR
U 10F9, 0001,003C,0180,42F8,0000,13F8             D[LONG]_CACHE         ;   GET 1ST COEF
                                        =;END

                                                  ;---------------------;
                                                  ID[T1]_D,LAB_R[R15],  ;   T1 GETS C0 <H>, LATCH TABLE ADDR
                                                  ALU_D, CHK.FLT.OPR,   ;   TEST C0 FOR RESERVED OP BEFORE SETTING FPD
U 13F8, 0001,003C,C580,3E7B,0800,1348             VA_VA+4               ;

                                        =00       ;00-------------------;
                                                  D[LONG]_CACHE,        ;   C0 <L>
                                                  LC_RC[T0],            ;   LATCH ARG <H>
U 1348, 0000,003D,0180,4100,0000,0E16             CALL,J/SETFPD         ;

                                        =10       ;10-------------------;   READ ERR
U 134A, 0014,0038,01C0,F800,0000,0EB8             Q_PC, J/BAKUP.PC      ;   BACK UP PC AND CAUSE EXCEPTION

                                                  ;11-------------------;
                                                  ID[T4]_D,             ;   SAVE C0 <L>
U 134B, 0010,0038,D180,3EA0,0000,13F9             R[R4]_LC              ;   R4 STORES ARG <H>
                                        =;END
                                                  ;---------------------;
U 13F9, 0000,003C,0180,FA98,0000,1238             R[R3]_LA              ;   R3 GETS TABLE ADDR
```

```
                                   ;13743  =0        ;-----------------------;
                                   ;13744            ALU_0(A),               ;    CLR RUNNING PROD <L>
                                   ;13745            LC_R[T2]&R1_ALU,        ;    LATCH ARG <L>
U 1238, 0003,003D,0180,FB90,0000,13E8  ;13746            CALL,J/POLY.PC          ;    GOTO GET PC DELTA
                                   ;13747
                                   ;13748            ;-----------------------;
                                   ;13749            SC_FE,                  ;    SC GETS BACK ARG EXP
U 1239, 001D,0030,0180,FA90,0081,13FA  ;13750            R[R2]_D.OR.Q            ;    R2 STORES PC DELTA, DEGREE
                                   ;13751  =;END
                                   ;13752            ;-----------------------;
                                   ;13753            R[R5]_LC,               ;    R5 STORES ARG <L>
U 13FA, 0010,0038,C5F0,2EA8,0000,13FB  ;13754            Q_ID[T1]                ;    GET C0 <H>
                                   ;13755
                                   ;13756            ;-----------------------;
                                   ;13757            FE_Q(EXP),CLK.UBCC,     ;    FE GETS C0 EXP
U 13FB, 0C01,203C,C1F0,2C00,0118,73FC  ;13758            D_Q,Q_ID[T0]            ;    GET BACK DEGREE
                                   ;13759
                                   ;13760            ;-----------------------;
                                   ;13761            R[R0]_D,                ;    R0 STORES C0 <H>
U 13FC, 0C01,003C,D1F0,2E80,0000,13FE  ;13762            D_Q,Q_ID[T4]            ;    D GETS DEGREE, Q GETS C0 <L>
                                   ;13763
                                   ;13764            ;-----------------------;
                                   ;13765            R[R1]_Q,                ;    STORE C0 <L> AS RUNNING SUM <L>
U 13FE, 0101,203C,0180,FA88,0000,140E  ;13766            D_D.LEFT2,SI/ZERO       ;    DEGREE * 4
                                   ;13767
                                   ;13768            ;-----------------------;
                                   ;13769            R[R3]&VA_LA+K[.8],      ;    SET UP TABLE ADDR IN R3, VA
                                   ;13770            D_D.LEFT,SI/ZERO,       ;    DEGREE * 8
                                   ;13771            SGN/CLR.SD+SS,          ;    CLEAR SS FOR EALU BRANCH
U 140E, 0518,0D14,0187,FA98,0200,1074  ;13772            D.NE.0?                 ;    DEGREE .NE. 0?
                                   ;13773
                                   ;13774  =10*      ;0----------------------;    DEGREE .EQ. 0
U 1074, 0000,123C,0180,F800,0000,1092  ;13775            EALU.Z?,J/POLYD.4       ;    IS C0 = 0?
                                   ;13776
                                   ;13777            ;1----------------------;    DEGREE > 0
U 1076, 0000,143C,0180,F800,0000,1351  ;13778            SC.GT.0?                ;    ARG = 0?
                                   ;13779  =;END
                                   ;13780
                                   ;13781  =*01      ;0----------------------;    ARG = 0
                                   ;13782            R[R3]_LA+D, VA_ALU,     ;    ADVANCE STATE TO LAST COEFFICIENT
U 1351, 001C,2014,0180,FA98,0200,1412  ;13783            J/POLYD.8               ;
                                   ;13784
                                   ;13785            ;1----------------------;    ARG .NE. 0
U 1353, 0800,003C,0180,FA20,0000,1413  ;13786            D_R[R4],J/POLYD.10      ;    ARG <H>
                                   ;13787  =;END
                                   ;13788
```

```
                                              =*01*
                                              POLYD.4:;0----------------------;   CO .NE. 0
U 1092, 0003,003C,0180,FAA0,0000,143C                 R[R4]_0, J/POLYD.35     ;   CLEAR R4, GO CLR R5 & R2 & EXIT

                                                      ;1----------------------;   CO .EQ. 0
U 1096, 0003,003C,0180,FA80,0000,1411                 R[R0]_0                 ;   ZERO RESULT<H>

                                                      ;-----------------------;
U 1411, 0003,003C,0180,FA88,0000,1092                 R[R1]_0, J/POLYD.4      ;   ZERO RESULT<L> AND GO CLEAN UP


                                              POLYD.8:;-----------------------;   CONTINUATION OF ARG=0 SPECIAL CASE CODE
U 1412, 0018,8038,0580,FA90,0000,1366                 R[R2]_K[.1], DT/BYTE    ;   TABLE ADDR & DEGREE NOW INDICATE LAST COEF
                                                                              ;   FALL INTO MAIN LOOP FOR 1 ITERATION

                                              ;       POLYD LOOP BEGINS HERE

                                              =110
                                              POLYD.9:;0----------------------;   NO:  NO INTERRUPTS
U 1366, 0800,003C,0180,FA20,0000,1413                 D_R[R4],J/POLYD.10      ;   ARG <H>

                                                      ;1----------------------;   YES:  INTERRUPTS
U 1367, 0000,003C,0180,F800,0000,121C                 J/POLY.INT              ;   INTRUPT ENTRY TO CLR TP AND BACKUP PC,R'S
                                              =;END

                                              POLYD.10:;----------------------;
U 1413, 0001,003C,0180,F980,0000,141E                 RC[T0]_D                ;   ARG <H>

                                                      ;-----------------------;
U 141E, 0000,003C,01C0,FA28,0000,1422                 Q_R[R5]                 ;   ARG <L>

```

```
                                         ;13820             ;-----------------------;
U 1422, 0800,003C,0180,FA00,0000,1423    ;13821             D_R[R0]                 :  PART PROD <H>
                                         ;13822
                                         ;13823             ;-----------------------;
                                         ;13824             ALU_D, STATE_K[.1],     :  SET POLYD FLAG FOR MULD
U 1423, 0001,003C,0580,FB08,1404,7280    ;13825             LAB_R1&RC[T1]_ALU       :  RC 1 GETS PARTIAL PROD <H>
                                         ;13826
                                         ;13827   =0**00    ;-----------------------;
                                         ;13828             D_Q,SC_K[.10],          :  D GETS PROD <L>
                                         ;13829             RC[T6]_LA,              :  SETUP FOR MULD
U 1280, 0C00,003D,6580,F9B0,0084,72C8    ;13830             CALL,J/MULD.02          :  CALL MULD RTN WITH STATE=1 AS FLAG
                                         ;13831
                                         ;13832   =1**00    ;00---------------------;  PARTIAL PROD = 0
                                         ;13833             D[LONG]_CACHE,          :  GET NEXT COEF <H>
                                         ;13834             LAB_R[R3],              :  LATCH TABLE ADDR
U 1290, 0000,003C,0180,4218,0000,142A    ;13835             J/POLYD.12              :
                                         ;13836
                                         ;13837   =1**10    ;10---------------------;  .25 <= PARTIAL PROD FRACT < .5
                                         ;13838             ALU_0+K[.1], Q_Q.LEFT,  :  SHIFT FRACTION 1 BIT TO THE LEFT
                                         ;13839               D_D.LEFT, SI/DIVD,    :
U 1292, 051B,0014,0428,F800,0084,B293    ;13840                   SC_SC-K[.1]       :  AND ADJUST EXPONENT TO MATCH
                                         ;13841
                                         ;13842             ;11---------------------;  .5 <= PARTIAL PROD FRACT <1.0
                                         ;13843             ID[T0]_D, RC[T3]_D,     :  SAVE PROD FRACT<H>
                                         ;13844               D_Q, Q_D,             :  SWAP HALVES SO WE CAN SAVE LO PART
                                         ;13845                 FE_SC,              :  SAVE EXP
U 1293, 0C01,143C,C1E0,3D98,0100,1371    ;13846                   SC?, J/POLYD.13   :  TEST FOR MULTIPLY OVERFLOW
                                         ;13847   =;END
                                         ;13848
                                         ;13849   POLYD.12:;--------------------;  PARTIAL PROD = 0
                                         ;13850             ALU_D, Q_ALU,           :  MOVE RESULT<H> TO Q THRU ALU
                                         ;13851             CHK.FLT.OPR,            :  CHECK FOR -0
                                         ;13852             EALU_D(EXP),CLK.UBCC,   :  CHECK FOR EXP 0
                                         ;13853             SGN/CLR.SD+SS,          :  CLEAR SS FOR EALU BRANCH
U 142A, 0001,003C,01C7,F803,0818,742B    ;13854             VA_VA+4                 :  INCREMENT TABLE ADDR
                                         ;13855
                                         ;13856             ;-----------------------;
U 142B, 0000,003C,0180,4000,0000,142C    ;13857             D[LONG]_CACHE           :  RESULT <L> - DON'T BUMP R3 HERE, MAY FAULT
                                         ;13858
                                         ;13859             ;-----------------------;
                                         ;13860             R[R3]_LA+K[.8],         :  INCREMENT TABLE ADDR TO POINT TO NEXT COEF
U 142C, 0018,1214,0180,FA98,0000,10EA    ;13861             EALU.Z?                 :  RESULT = 0?
                                         ;13862
                                         ;13863   =*01*     ;0----------------------;  NOT 0
                                         ;13864             R[R1]_D,                :  STORE RESULT <L>
U 10EA, 0001,003C,0180,FA88,0000,1253    ;13865             J/POLYD.21              :  GO STORE RESULT<H>
                                         ;13866
                                         ;13867             ;1----------------------;  RESULT 0
                                         ;13868             R[R0]_0, Q_0, D_0,      :  RESULT IS A TRUE 0
U 10EE, 0F03,003C,01F8,FA80,0000,1253    ;13869             J/POLYD.21              :
                                         ;13870   =;END
                                         ;13871
```

```
                                          =001                            ;  TEST FOR MULTIPLY OVERFLOW
                                          POLYD.13:;00--------------------;  PARTIAL PROD .NE. 0, EXP = 0
                                                   ID[T2]_D, ALU_Q,       ;  SAVE PROD FRAC<L>,
                                                    D_ALU.LEFT, SI/ZERO,  ;  PUT PROD FRAC<H> IN D NORMALIZED
U 1371, 0821,203C,C980,3E18,0000,142D               [AB_R[R3], J/POLYD.14 ;  LATCH COEF TABLE ADDRESS

                                                   ;01--------------------;  PROD .NE. 0,  0 < EXP < 100
                                                   ID[T2]_D, ALU_Q,       ;  SAVE PROD FRAC<L>,
                                                    D_ALU.LEFT, SI/ZERO,  ;  PUT PROD FRAC<H> IN D NORMALIZED
U 1373, 0821,203C,C980,3E18,0000,142D               [AB_R[R3], J/POLYD.14 ;  LATCH COEF TABLE ADDRESS

                                                   ;10--------------------;  PROD .NE. 0, EXP < 0
                                                   ID[T2]_D, ALU_Q,       ;  SAVE PROD FRAC<L>,
                                                    D_ALU.LEFT, SI/ZERO,  ;  PUT PROD FRAC<H> IN D NORMALIZED
U 1375, 0821,203C,C980,3E18,0000,142D               [AB_R[R3], J/POLYD.14 ;  LATCH COEF TABLE ADDRESS

                                                   ;11--------------------;  PROD .NE. 0, EXP > 100 (OVERFLOW)
U 1377, 0000,003C,7580,F800,1404,7371              STATE_K[.20], J/POLYD.13, SET FLAG SAYING THAT PRODUCT OVERFLOWED.
                                                                          ;  THIS FLAG IS NECESSARY BECAUSE ADDD
                                                                          ;  WILL LOSE TRACK OF THE 9TH EXPONENT BIT
                                                                          ;  DURING ITS CALCULATIONS (SIGH)

                                          POLYD.14:;----------------------;
                                                   EALU_FE,RC[T0]_PACK.FP, ;  SAVE PACKED PROD<H>, IGNORING POSS. UNDRFLOW
U 142D, 0008,0038,0180,4180,0000,7430              D[LONG]_CACHE          ;  READ NEXT COEF<H>

                                                   ;----------------------;
                                                   RC[T1]_D, CHK.FLT.OPR, ;  CHECK FOR -0
U 1430, 0001,003C,01E0,F98B,0800,1431              Q_D, VA_VA+4           ;  INCREMENT TABLE ADDR

                                                   ;----------------------;
U 1431, 0000,003C,0180,40C[illegible],0000,1432              D[LONG]_CACHE          ;  READ COEF<L> - DON'T BUMP R3 YET, MAY FAULT

                                                   ;----------------------;
U 1432, 0018,0014,0180,FA98,0000,1358              R[R3]_LA+K[.8]         ;  INCREMENT TABLE ADDR
```

```
                                          ;13907  =00       ;00----------------------------; CALL CONSTRAINT BLOCK
                                          ;13908            STATE_STATE.OR.K[.1],          ; SET POLYD FLAG AGAIN FOR ADDD
U 1358, 0001,003D,0580,F9B0,1404,28BE    ;13909            RC[T6]_D,CALL,J/UNPK           ; SAVE COEF<L> AND UNPACK COEF.
                                          ;13910
                                          ;13911            ;01----------------------------; IT IS 0
                                          ;13912            SD_SS,Q_ID[T2],                ; PROD <L>
U 1359, 0000,003C,C9F4,2C00,0000,1433    ;13913            J/POLYD.16                     ;
                                          ;13914
                                          ;13915            ;1J----------------------------; NOT 0
                                          ;13916            R[R15]_K[SC],                  ; SAVE SC WHILE WE COMPUTE EXP DIFF
                                          ;13917            FE_SC-FE,SC_SC-FE,             ; GET EXPONENT DIFFERENCE
U 135A, 0018,0038,1D80,FAF8,0190,B241    ;13918            CLR.UBCC                       ; FOR COMPARE EXP'S
                                          ;13919  =
                                          ;13920
                                          ;13921  =0*001    ;0*001-------------------------; CALL SITE FOR ADDD
                                          ;13922            ALU_0-K[SC],SC_ALU,            ; GET NEG OF EXP DIFF IN CASE ITS POS
                                          ;13923            LAB_R[R15],                    ; LATCH SAVED SC FOR RELOADING
U 1241, 001B,1401,1D80,FA78,0082,1395    ;13924            SC.GT.0?,CALL[POLYD.A]         ; GO ADD, NEGATING EXP DIFF IF IT IS POS
                                          ;13925
                                          ;13926
                                          ;13927
                                          ;13928  =1*001    ;1*001-------------------------; SC.eq.0, UNDERFLOW OR OVERFLOW
U 1251, 0000,163C,0180,F800,0000,10B5    ;13929            STATE5?,J/POLYD.15             ; DEPENDING ON OV FLAG
                                          ;13930
                                          ;13931  POLYD.21:
                                          ;13932            ;1*011-------------------------; SC.eq.[01 to FF], OK OR OVERFLOW
U 1253, 0000,163C,0180,F800,0000,123C    ;13933            STATE5?,J/POLYD.17             ; OVERFLOW, DEPENDING ON OV FLAG
                                          ;13934
                                          ;13935            ;1*101-------------------------; SC.lss.0, UNDERFLOW OR OK
U 1255, 0000,163C,0180,F800,0000,1081    ;13936            STATE5?,J/POLYD.24             ; UNDERFLOW OR OK, DEPENDING ON OV FLAG
                                          ;13937
                                          ;13938            ;1*111-------------------------; SC.gt.FF, OVERFLOW
                                          ;13939            D_K[.8],                       ; D GETS OVERFLOW TRAP CODE
U 1257, 0818,0038,0180,FA18,0000,13F1    ;13940            LAB_R[R3],J/POLYD.FAULT        ; FETCH TABLE ADDRESS
```

```
                                        ;13941  =101
                                        ;13942  POLYD.A:;101-------------------; SUBR TO TALK TO ADDD - EXP DIFF <=0 ENTRY
                                        ;13943          LC_RC[T1], SC_LA,       ; LATCH PACKED COEF<H>, SET EXP = DST EXP,
                                        ;13944          Q_ID[T2],               ; PUT PROD<L> IN Q WHERE ADDD EXPECTS IT
U 1395, 0000,123C,C9F0,2D08,0082,0182   ;13945          EALU?, J/ADDD.10        ; DISPATCH INTO THE ADDD ROUTINE
                                        ;13946
                                        ;13947          ;1---------------------; EXP DIFF > 0 ENTRY
                                        ;13948          LC_RC[T1], SC_LA,       ; LATCH PACKED COEF<H>, SET EXP=DST EXP,
                                        ;13949          Q_ID[T2],               ; PUT PROD<L> IN Q WHERE ADDD EXPECTS IT
U 1397, 0000,123C,C9F0,2D08,0182,0182   ;13950          FE_SC, EALU?, J/ADDD.10 ; NEGATE DIFF & DISPATCH INTO THE ADDD ROUTINE
                                        ;13951
                                        ;13952
                                        ;13953  POLYD.16:;---------------------; COME HERE WHEN COEFFICIENT = 0
                                        ;13954          D_Q,Q_RC[T3],           ; SET FOR PACKING DOUBLE RESULT
                                        ;13955          SC_FE,                  ; SC GETS EXP
U 1433, 0C10,0038,7DC0,F918,0185,7438   ;13956          FE_K[.18]               ; SET 24. FOR PACKING DOUBLE RESULT
                                        ;13957
                                        ;13958          ;----------------------;
                                        ;13959          D_D.LEFT,Q_Q.LEFT,      ; DOUBLE SHIFT <Q,D>
U 1438, 0500,003C,4128,F800,0000,10A1   ;13960          SI/ASHL,K[.80]          ;
                                        ;13961
                                        ;13962  =00001  ;00001-----------------; ** THIS CONSTRAINT USED FOR MANY THINGS
                                        ;13963          D_Q,Q_D+K[.80],         ; RESULT FRACS IN <D,Q>
                                        ;13964          C[K.UBCC,               ; CHECK IF CARRY
U 10A1, 0C19,0015,41C0,F800,0010,03FC   ;13965          CALL,J/PACKD            ; CALL PACK SUBRT
                                        ;13966  =10001
                                        ;13967  POLYD.24:
                                        ;13968          ;10001-------------------------; RESULT UNDERFLOW
U 10B1, 0000,003D,3DF0,2C00,0000,12AE   ;13969          Q_ID[PSL],CALL[PSLFU.A]        ; GO SEE IF PSL<fu> IS SET
                                        ;13970
                                        ;13971          ;10011-------------------------; RESULT OK
U 10B3, 0001,203C,01E0,FA80,0000,1439   ;13972          R[R0]_Q,Q_D,J/POLYD.22         ; STORE RESULT <H>, Q GETS RESULT <L>
                                        ;13973
                                        ;13974  POLYD.15:
                                        ;13975          ;10101-------------------------; RESULT UNDERFLOW
U 10B5, 0000,003D,3DF0,2C00,0000,12AE   ;13976          Q_ID[PSL],CALL[PSLFU.A]        ; GO SEE IF PSL<fu> IS SET
                                        ;13977
                                        ;13978          ;10111-------------------------; RESULT OVERFLOW
                                        ;13979          D_K[.8],                       ; D GETS OVERFLOW TRAP CODE
U 10B7, 0818,0038,0180,FA18,0000,13F1   ;13980          LAB_R[R3],J/POLYD.FAULT        ; FETCH TABLE ADDRESS
                                        ;13981
                                        ;13982  =11101  ;11101-------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
                                        ;13983          R[R0]_0,STATE_K[ZERO],Q_0,     ; PRETEND IT DIDN'T HAPPEN
U 10BD, 0003,003C,19F8,FA80,1404,7439   ;13984          J/POLYD.22                     ;
                                        ;13985
                                        ;13986          ;11111-------------------------; RETURN HERE IF PSL<fu>.eq.1(SC.ne.0)
                                        ;13987          D_K[.A],                       ; D GETS UNDERFLOW TRAP CODE
U 10BF, 0818,0038,F580,FA18,0000,13F1   ;13988          LAB_R[R3],J/POLYD.FAULT        ; FETCH TABLE ADDRESS
```

```
                                             ;13989  =**0*
                                             ;13990  POLYD.17:
                                             ;13991          ;**0*---------------------------; BRANCH CONSTRAINT FOR STATE<5>
                                             ;13992          R[R0]_Q,Q_D,                   ; SAVE HIGH RESULT, Q GETS LO RESULT
U 123C, 0001,203C,19E0,FA80,1404,7439        ;13993          STATE_K[ZERO],J/POLYD.22       ; CLEAR ALL FLAGS
                                             ;13994
                                             ;13995          ;  ;*---------------------------;
                                             ;13996          D_K[.8],                       ; D GETS OVERFLOW TRAP CODE
U 123E, 0818,0038,0180,FA18,0000,13F1        ;13997          LAB_R[R3],J/POLYD.FAULT        ; FETCH TABLE ADDRESS
                                             ;13998
                                             ;13999  POLYD.22:
                                             ;14000          ;-------------------------------;
U 1439, 0000,003C     ).FA10,0000,143A       ;14001          LAB_R[R2]                      ;
                                             ;14002
                                             ;14003  POLYD.26:
                                             ;14004          ;-------------------------------;
                                             ;14005          VA_VA+4,                       ; INC COEF ADDR
                                             ;14006          R[R2]_LA-K[.1],DT/BYTE,        ; DECREMENT DEGREE COUNT
U 143A, 0018,8000,0580,FA93,0010,143B        ;14007          CLK.UBCC                       ; SET Z IF WE ARE DONE
                                             ;14008
                                             ;14009          ;-------------------------------;
U 143B, 0001,213C,0180,FA88,4000,124C        ;14010          R[R1]_Q,INTRPT.STROBE,Z?       ; STORE RESULT<L>, ENABLE INT,  DONE?
                                             ;14011
                                             ;14012  =0      ;0------------------------------; NO:
U 124C, 0000,0E3C,0180,F800,0000,1366        ;14013          INTERRUPT.REQ?,J/POLYD.9       ; GOTO NEXT LOOP
                                             ;14014
                                             ;14015          ;1------------------------------; YES:
U 124D, 0018,0038,1980,FAA0,0000,143C        ;14016          R[R4]_K[ZERO]                  ; CLR R4 (ARG <H>)
                                             ;14017
                                             ;14018  POLYD.35:
                                             ;14019          ;-------------------------------;
U 143C, 0018,0038,1980,FAA8,2000,143D        ;14020          CLR.FPD,R[R5]_K[ZERO]          ; CLR PSL <FPD> AND ARG<L>
                                             ;14021
                                             ;14022          ;-------------------------------;
                                             ;14023          R[R2]_K[ZERO],                 ; CLR DEGREE--UNDERFLOW FLAG--PC DELTA
                                             ;14024          SC_FE,                         ; GET BACK EXP
U 143D, 0018,003B,1980,FA90,0081,12C2        ;14025          SUB/SPEC,J/POLY.C              ; GOTO SET COND CODES
                                             ;14026
                                             ;14027  .REGION/0000,0FFF                      ;Lower 4k for PCS
```

```
14027; This page intentionally left blank.
```

```
                                        ;14028  .TOC    "INIT2.MIC"
                                        ;14029  .TOC    "Revision 0.2"
                                        ;14030  ;        P. R. Guilbault
                                        ;14031
;14032  .NOBIN
;14033  .TOC    "       Revision History"
;14034
;14035  ; 00    Start of history
;14036
                                        ;14037  .BIN
                                        ;14038  .NOLIST         ;Disable listing of PCS code for quickie assemblies
```

```
                                          .TOC "  Initialize microcode  :INITIALIZE MACHINE ROUTINE"

                                                  ;************************
                                                  ;                       *
                                                  ; INITIALIZE MACHINE    *
                                                  ;                       *
                                                  ;************************

                                                  ;THIS ROUTINE IS USED TO INITIALIZE THE REQUIRED ID-REGISTERS,
                                                  ;TBUF, CACHE, AND OPTIONAL ACCELERATOR.  AFTER COMPLETING THESE
                                                  ;TASKS, THIS ROUTINE ENTERS A WAIT LOOP, WAITING FOR
                                                  ;THE CONSOLE TO INITIATE EITHER A WARM START OR COLD START
                                                  ;PROCEDURE.

                                                  ;INITIALIZE ID REGISTERS

                                          100:
                                          SYS.INIT:
                                                  ;-----------------------;
                                                  SC_K[.F],               ;  15 TO SC TO CONSTRUCT BIT MASK
U 0100, 1000,003C,6180,0800,0084,6984             STOP.IB, MCT/MEM.NOP    ;  STOP IBUFFER
                                                                          ;  CONSOLE MODE SET, SO IF ERROR GO TO IT

                                                  ;-----------------------;
                                                  D_NOT.MASK,      Q_0,   ;  A "1" TO D<15> POSITION
U 0984, 0803,001C,D5F8,F800,0084,6985             SC_K[.6]                ;  SETUP SC FOR LEFT SHIFT OF 6

                                                  ;-----------------------;
U 0985, 0D00,003C,0180,F800,0000,01E8             D_DAL.SC                ;  SHIFT A "1" BIT INTO POS<21>

                                          =0100*  ;0100*-----------------;
                                                  ID[MAINT]_D,            ;  SET SBI ENABLE INVALIDATE BIT
U 01E8, 0000,003D,7580,3C00,0000,08D0             CALL, J/WPR.10A         ;  GO AND CLEAR TBUF
```

```
                                        ;14072          ;*************************
                                        ;14073          ;                        *
                                        ;14074          ; CLEAR CACHE ROUTINE    *
                                        ;14075          ;                        *
                                        ;14076          ;*************************
                                        ;14077
                                        ;14078          ;THIS ROUTINE IS USED TO RESET THE CONTENTS OF CACHE.
                                        ;14079          ;THIS IS DONE BY ISSUING THE INVALIDATE CACHE MICRO ORDER
                                        ;14080          ;TO EACH AND EVERY CACHE TAG LOCATION (1024).  BOTH
                                        ;14081          ;GROUPS, G0 & G1, ARE RESET TOGETHER.
                                        ;14082
                                        ;14083  =1100*  ;1100*-----------------;
                                        ;14084          SC_K[ZERO],     D_0,    ;  CLEAR SC AND D-REGS
                                        ;14085          VA_K[ZERO],             ;  INITIALIZE VA TO ZERO
U 01F8, 0F18,0038,1980,F800,0284,64B7   ;14086          J/I.CACHE.5             ;  GO & WRITE INVALIDATES
                                        ;14087  =
                                        ;14088  =011
                                        ;14089  I.CACHE.3:
                                        ;14090          ;011-------------------;  *** LOOP FINISHED ***
                                        ;14091          D_0,                    ;  CLEAR D-REG
                                        ;14092          Q_ID[ACC.CS],           ;  GET ACCELERATOR'S CONTROL/STATUS REG
U 04B3, 0F00,003C,5DF0,2C00,0000,0988   ;14093          J/CLR.ACC               ;
                                        ;14094
                                        ;14095  I.CACHE.5:
                                        ;14096          ;111-------------------;  *** CONTINUE LOOP ***
                                        ;14097          CACHE.INVALIDATE,       ;  INVALIDATE G0 & G1 TAG ADDRESSED BY VA
U 04B7, 0000,003C,0180,9000,0080,C986   ;14098          SC_SC+1                 ;  INCREMENT COUNT
                                        ;14099
                                        ;14100          ;----------------------;
                                        ;14101          VA_VA+4,                ;  INCREMENT
U 0986, 0000,0C3C,0180,F803,0000,04B3   ;14102          SC.NE.0?,J/I.CACHE.3    ;  CHECK TO SEE IF ALL 1024 WRITTEN
```

```
                                        ;14103          ;*****************************
                                        ;14104          ;                            *
                                        ;14105          ; RESET ACCELERATOR ROUTINE *
                                        ;14106          ;                            *
                                        ;14107          ;*****************************
                                        ;14108
                                        ;14109          ;THIS ROUTINE CHECK TO SEE IF AN OPTIONAL ACCELERATOR
                                        ;14110          ;IS ON THIS MACHINE, AND IF SO ISSUES A RESET TO
                                        ;14111          ;IT.  IT THEN WAITS UNTIL THE ACCELERATOR ISSUES AN ACCELERATOR
                                        ;14112          ;SYNC, INDICATING THAT THE ACCELERATOR HAS INITIALIZED ITSELF.
                                        ;14113          ;THE ACCELERATOR CONTROL AND STATUS REGISTER IS ALSO RESET.
                                        ;14114
                                        ;14115  CLR.ACC:
                                        ;14116          ;-----------------------;
                                        ;14117          ALU_Q.AND.K[.F],        ;  TEST TO SEE IF THERE IS AN ACCELERATOR
U 0988, 0019,2034,6180,F800,0010,0989   ;14118          CLK.UBCC                ;  CLOCK ALU<Z> FOR TEST
                                        ;14119
                                        ;14120          ;-----------------------;
                                        ;14121          ID[ACC.CS]_D,           ;  RESET ACCELERATOR'S CONTROL & STATUS REG
U 0989, 0000,013C,5D80,3C00,0000,05AC   ;14122          Z?                      ;  IS AN ACCELERATOR ATTACHED?
                                        ;14123
                                        ;14124  =0      ;0----------------------;  *** ACCELERATOR IS INSTALLED ***
                                        ;14125          TRAP.ACC[1],            ;  TRAP ACCELERATOR TO ITS INITIALIZATION
U 05AC, 0000,00BC,0080,F800,0000,04C6   ;14126          J/I.ACC.05              ;  ROUTINE
                                        ;14127
                                        ;14128          ;1----------------------;  *** NO ACCELERATOR ****
U 05AD, 0000,003C,0180,F800,0000,098A   ;14129          J/I.ACC.07              ;
                                        ;14130  =110
                                        ;14131  I.ACC.05:
                                        ;14132          ;110--------------------;  *** NO ***
                                        ;14133          D_K[.8000],             ;  SETUP ENABLE BIT FOR ACCEL
U 04C6, 0818,0638,4580,F800,0000,04C6   ;14134          ACC.SYNC?,  J/I.ACC.05  ;  HAS ACCELERATOR INITIALIZED ITSELF?
                                        ;14135
                                        ;14136          ;111--------------------;  *** YES ***
U 04C7, 0000,003C,5D80,3C00,0000,098A   ;14137          ID[ACC.CS]_D            ;  ENABLE ACCELERATOR
```

```
                                          I.ACC.07:;-----------------------;
U 098A, 0818,0038,1180,F800,0000,098C             D_K[.4]                 ;   VALUE TO SET ASTLVL

                                                  ;-----------------------;
U 098C, 0000,003C,3180,3C00,0000,098E             ID[CES]_D               ;   SET ASTLVL TO 4

                                                  ;-----------------------;
U 098E, 0818,0038,9180,F800,0000,0994             D_K[.1F00]              ;   GET 1F TO SET IPL TO 31

                                                  ;-----------------------;
U 0994, 0819,0014,1180,F800,0000,0996             D_D+K[.4]               ;   BIT TO SET 'IS'

                                                  ;-----------------------;
                                                  D_D.SWAP,               ;   FINISH SETUP OF NEW PSL
U 0996, 0818,0038,1980,F801,0200,0998             PC&VA_K[ZERO]           ;   RESET PC TO ZERO

                                                  ;-----------------------;
U 0998, 0000,003C,3D80,3C00,0000,0999             ID[PSL]_D               ;   SET IPL=31 AND IS=1


                                          ;SETUP CONSTANT FOR MASKING OUT PTE MBZ FIELD IN GETPTE ROUTINE
                                                  ;-----------------------;
U 0999, 081B,001C,9980,F800,0000,099C             D_NOT.K[.E003]          ;   GET CONSTANT FOR PTE MBZ FIELD

                                                  ;-----------------------;
U 099C, 0B00,003C,0180,F800,0000,099E             D_D.SWAP                ;   ALIGN CONSTANT WITH MBZ FIELD

                                                  ;-----------------------;
U 099E, 0001,003C,0180,F9F8,0000,09A0             RC[PTE.MASK]_D          ;   SETUP AND LEAVE IN RC REG FOR LATER USE

                                                  ;-----------------------;
                                                  D_K[.3],                ;   WARM/COLD START CODE TO D-REG
U 09A0, 0818,0038,0D80,F800,0000,039F             J/HALT.ERR              ;   HALT MACHINE

                                          .LIST           ;Re-enable full listing
```

```
14172; This page intentionally left blank.
```

```
                                        ;14173  .TOC    "ASPC.MIC"
                                        ;14174  .TOC    "Revision 1.1"
                                        ;14175  ;        P. R. Guilbault
                                        ;14176
;14177  .NOBIN
;14178  .TOC    "       Revision History"
;14179
;14180  ; 01    Remove absolute jumps.
;14181  ; 00    Start of history
;14182
                                        ;14183  .BIN
                                        ;14184  .NOLIST         ;Disable listing of PCS code for quickie assemblies
```

```
                                            .TOC    "       I-stream decode forks : Address Specifier Evaluation"

                                            ; Control passes to this point by CALL.J/ASPC and returns to the call site "or"
                                            ; 60, if the specifier is valid, or the call site "or" 61 if the specifier is
                                            ; register mode, which will be invalid unless the specifier is a field source.
                                            ; At the return, Q contains the longword that was in D at the entry, D and VA
                                            ; contain the address implied by the specifier.

                                            47E:     ;---------------------------------;HANG HERE IF IB MUST STALL
                                            ASPC:    Q_IB.DATA,                         ;GET SPECIFIER DATA FROM ISTREAM
                                                     LAB_R(SP1),                        ;LOAD LATCHES FROM BASE SPECIFIER
                                                     CLR.IB.COND,                       ;DISCARD BASE OPERAND SPECIFIER
                                                     PC_PC+N,                           ;STEP PC OVER IT
                                                     MCT/ALLOW.IB.READ,                 ;LET IB GET NEEDED DATA
U 047E, F000,003F,01F0,F847,0000,0400                SUB/SPEC.J/ASPC.B                  ;EVALUATE THE SPECIFIER

                                            400:     ;---------------------------------;
U 0400, 0000,003C,0180,F800,0000,0001       ASPC.B:  J/RSVMOD                           ;S^# SHORT LITERAL NOT LEGAL AS ASRC

                                            401:     ;---------------------------------;
U 0401, 0000,003C,0180,F800,0000,0001                J/RSVMOD                           ;RESERVED MODE

                                            402:     ;---------------------------------;
U 0402, 0000,003C,0180,F800,0000,0001                J/RSVMOD                           ;QUAD/DOUBLE SHORT LITERAL

                                            403:     ;---------------------------------;
U 0403, 0000,003C,0180,F800,0000,0001                J/RSVMOD                           ;RESERVED MODE

                                            404:     ;---------------------------------;
                                                     Q_D,                               ;REGISTER
                                                     R(PRN)_LA+K[ZERO].RLOG,            ;RECORD REGISTER NUMBER
                                                     D_ALU,                             ;GET REGISTER CONTENTS TO D
U 0404, 0818,001A,19E0,F8D8,0000,0061                RETURN61                           ;RETURN IT IN CASE FIELD SOURCE

                                            424:     ;---------------------------------;
                                                     Q_D,                               ;WRITE REGISTER
                                                     R(PRN)_LA+K[ZERO].RLOG             ;RECORD REGISTER NUMBER
                                                     D_ALU,                             ;GET REGISTER CONTENTS TO D
U 0424, 0818,001A,19E0,F8D8,0000,0061                RETURN61                           ;RETURN IT IN CASE FIELD SOURCE

                                            405:     ;---------------------------------;
U 0405, 0000,003C,0180,F800,0000,0001                J/RSVMOD

                                            406:     ;---------------------------------;
                                                     Q_D,                               ;QUAD REGISTER
                                                     R(PRN)_LA+K[ZERO] RLOG,            ;RECORD REGISTER NUMBER
U 0406, 0818,001A,19E0,F8D8,0000,0061                D_ALU,RETURN61                     ;RETURN CONTENTS TO CALLER

                                            426:     ;---------------------------------;
                                                     Q_D,                               ;WRITE QUAD REGISTER
                                                     R(PRN)_LA+K[ZERO].RLOG,            ;RECORD REGISTER NUMBER
U 0426, 0818,001A,19E0,F8D8,0000,0061                D_ALU,RETURN61                     ;RETURN CONTENTS TO CALLER

                                            407:     ;---------------------------------;
U 0407, 0000,003C,0180,F800,0000,0001                J/RSVMOD                           ;ILLEGAL QUAD REG
```

```
                                        ;14240  ;ADDRESS SPECIFIER EVALUATION, CONTINUED
                                        ;14241
                                        ;14242  408:     ;---------------------------------------;
                                        ;14243  ASPC.DR:Q_D,D&VA_LA,                            ;(R)
                                        ;14244           LC_RC[T7],                             ;RECOVER INDEX, IF ANY
U 0408, 0800,003E,01E0,F938,0200,0060   ;14245           RETURN60                               ;RETURN IT
                                        ;14246
                                        ;14247  409:     ;---------------------------------------;
                                        ;14248           R(PRN)_LA+K[SP1.CON].RLOG,             ;UPDATE THE STACK POINTER
U 0409, 0018,0018,1580,F8D8,0000,0408   ;14249           J/ASPC.DR                              ;THEN LOAD UN-INCREMENTED ADDR
                                        ;14250
                                        ;14251  40A:     ;---------------------------------------;
                                        ;14252           Q_D,R(PRN)_LA-K[SP1.CON].RLOG,         ;-(R) AUTO DECREMENT
                                        ;14253           D_ALU,                                 ; USE DECREMENTED ADDR
U 040A, 0818,0004,15E0,F8D8,0000,09A6   ;14254           J/ASPC.DF                              ;GO LOAD LC BEFORE RETURNING
                                        ;14255
                                        ;14256  40B:     ;---------------------------------------;
U 040B, 0000,003C,01E0,F800,0200,09A1   ;14257           Q_D,VA_LA                              ;@(R)+ AUTO INCREMENT DEFERED
                                        ;14258
                                        ;14259           ;---------------------------------------;
                                        ;14260           D[LONG]_CACHE,                         ;GET INDIRECT WORD
                                        ;14261           R(PRN)_[A+K[.4].RLOG,                  ; WHILE UPDATING REGISTER
U 09A1, 0018,0018,1180,40D8,0000,09A6   ;14262           J/ASPC.DF                              ;THEN JOIN COMMON CODE
                                        ;14263
                                        ;14264  40C:     ;---------------------------------------;
                                        ;14265           RC[T7]_LA.CTX,                         ;INDEX MODE, CONTEXT SHIFT INDEX
U 040C, 0060,C03D,0180,F9B8,0000,047E   ;14266           CALL,J/ASPC                            ;GO AROUND AGAIN
                                        ;14267
                                        ;14268  46C:     ;---------------------------------------;
                                        ;14269           D&VA_D+LC,                             ;RETURN THE INDEXED VALUE
U 046C, 0811,0016,0180,F800,0200,0060   ;14270           RETURN60
                                        ;14271
                                        ;14272  40D:     ;---------------------------------------;
                                        ;14273           Q_D,D&VA_Q+LB.PC,                      ;D(R) DISPLACEMENT MODE.
                                        ;14274           C[R.IB.SPEC,                           ;DISCARD THE SPECIFIER
                                        ;14275           LC_RC[T7],                             ;LOAD UP INDEX, IF ANY
U 040D, D805,2016,01E0,F938,0200,0060   ;14276           RETURN60
                                        ;14277
                                        ;14278  40F:     ;---------------------------------------;
                                        ;14279           Q_D,VA_Q+LB.PC,                        ;@D(R) DISPLACEMENT DEFERED
U 040F, D005,2014,01E0,F800,0200,09A4   ;14280           C[R.IB.SPEC                            ;DROP THE SPECIFIER
                                        ;14281
                                        ;14282           ;---------------------------------------;
U 09A4, 0000,003C,0180,4000,0000,09A6   ;14283           D[LONG]_CACHE                          ;GET INDIRECT, GO USE IT AS ADDR
                                        ;14284
                                        ;14285           ;---------------------------------------;
                                        ;14286  ASPC.DF:VA_D,                                   ;USE POINTER AS ADDRESS
                                        ;14287           LC_RC[T7]                              ;RECOVER INDEX, IF ANY
U 09A6, 0001,003E,0180,F938,0200,0060   ;14288           RETURN60                               ;RETURN IT
```

```
                                        ;14289  ;HERE ARE VARIANTS OF THE ASPC ENTRY POINTS FOR R=PC
                                        ;14290
                                        ;14291  414:    ;--------------------------------;
U 0414, 0000,003C,0180,F800,0000,0001   ;14292          J/RSVMOD                          ;PC REGISTER MODE
                                        ;14293
                                        ;14294  415:    ;--------------------------------;
U 0415, 0000,003C,0180,F800,0000,0001   ;14295          J/RSVMOD                          ;ILLEGAL REGISTER MODE, R=PC
                                        ;14296
                                        ;14297  416:    ;--------------------------------;
U 0416, 0000,003C,0180,F800,0000,0001   ;14298          J/RSVMOD                          ;PC QUAD REGISTER MODE
                                        ;14299
                                        ;14300  417:    ;--------------------------------;
U 0417, 0000,003C,0180,F800,0000,0001   ;14301          J/RSVMOD                          ;ILLEGAL QUAD REGISTER MODE, R=PC
                                        ;14302
                                        ;14303  418:    ;--------------------------------;
U 0418, 0000,003C,0180,F800,0000,0001   ;14304          J/RSVMOD                          ;(PC)
                                        ;14305
                                        ;14306  419:    ;--------------------------------;
                                        ;14307          Q_D,                              ;(PC)+ IMMEDIATE MODE
                                        ;14308          D_PC,                             ;GET PC WHICH THE IB INCREMENTED
U 0419, D814,0038,01E0,F800,0000,09A8   ;14309          CLR.IB.SPEC
                                        ;14310
                                        ;14311          ;--------------------------------;
                                        ;14312          D&VA_D-K[SP1.CON],                ;COMPUTE THE UNINCREMENTED ADDRESS
                                        ;14313          LC_RC[T7],                        ;RECOVER INDEX, IF ANY
U 09A8, 0819,0002,1580,F938,0200,0060   ;14314          RETURN60
                                        ;14315
                                        ;14316  41A:    ;--------------------------------;
U 041A, 0000,003C,0180,F800,0000,0001   ;14317          J/RSVMOD                          ;-(PC)
                                        ;14318
                                        ;14319  41B:    ;--------------------------------;
                                        ;14320          Q_D,D&VA_Q,                       ;@(PC)+ ABSOLUTE MODE
                                        ;14321          LC_RC[T7],                        ;GET BACK INDEX
                                        ;14322          CLR.IB.SPEC,
U 041B, DC01,203E,01E0,F938,0200,0060   ;14323          RETURN60
                                        ;14324
                                        ;14325  41C:    ;--------------------------------;
U 041C, 0000,003C,0180,F800,0000,0001   ;14326          J/RSVMOD                          ;INDEX MODE, R=PC
                                        ;14327
                                        ;14328  41D:    ;--------------------------------;
U 041D, 0000,003C,0180,F800,0000,0001   ;14329          J/RSVMOD                          ;NESTED INDEX MODE, R=PC
```

```
                                               ;HERE WHEN IB ROMS BELIEVE WE SHOULDN'T GET HERE

                                               487:     ;--------------------------------;
U 0487, 0000,C03D,0180,F800,0000,0EE0                   CALL,J/EH.USEQ                   ;''UNUSED'' LOCATION IN IB ROM

                                               ;HERE OFF ASPC, WHEN INSTRUCTION BUFFER DOES NOT HAVE ENOUGH DATA

                                               47C:     ;--------------------------------;
U 047C, 0000,003D,0180,F800,0000,0E64                   CALL,J/IB.TBM                    ;TB MISS.  REFILL IT

                                               47D:     ;--------------------------------;
U 047D, 0000,003D,0180,F800,0000,0B80                   CALL,J/IB.ERR                    ;ANY ERROR.  FIND OUT WHAT HAPPENED

                                               ;47E:    ;--------------------------------;
                                               ;        ASPC:                            ;STALL, WAITING FOR THE DATA

                                               47F:     ;--------------------------------;INTERRUPT REQUEST DETECTED
U 047F, 0000,003C,0180,F800,0000,04F8                   J/INT.B                          ;BACKUP REGISTERS AND TAKE INTERRUPT

                                               =( *     ;--------------------------------;
                                               INT.B:   RC[PC.SV]_PC,                    ;SAVE PC WHERE BAKUP.PC WILL FIND IT
U 04F8, 0014,1539,0180,F9E0,0000,0DA7                   CALL,RLOG.EMPTY?,J/BAKUP.RGS     ;GO RESTORE REGISTERS AND PC

                                                        ;--------------------------------;HERE ON INTERRUPT IN MIDST OF INSTR
                                               INT.I:   Q_ID[PSL],                       ;GET CURRENT PSL
U 04FA, 0883,0028,3DF0,2C00,0000,09A9                   ALU_-1,D_ALU.RIGHT2,SI/ZERO      ;BUILD MASK FOR BITS 29:0

                                               A .PA.30:
                                                        ;--------------------------------;
U 09A9, 081D,0034,0180,F800,0000,09AA                   D_D.AND.Q                        ;CLEAR TP (CM =0 ANYWAY)

                                                        ;--------------------------------;
                                                        ID[PSL]_D,                       ;PUT BACK PSL
U 09AA, 0000,0E3C,3D80,3C00,0000,0F8D                   INTERRUPT.REQ?,J/INTIO

                                               .LIST             ;Re-enable full listing
```

```
14365; This page intentionally left blank.
```

```
                                        ;14366  .TOC     "FIELD.MIC"
                                        ;14367  .TOC     "Revision 1.1"
                                        ;14368  :         P. R. Guilbault
                                        ;14369
;14370  .NOBIN
;14371  .TOC     "       Revision History"
;14372
;14373  ; 01     Remove absolute jumps.
;14374  ; 00     Start of history
;14375
                                        ;14376  .BIN
                                        ;14377  .NOLIST          ;Disable listing of PCS code for quickie assemblies
```

```
                                          .TOC    "       Field instructions    : FFS, FFC, CMPV, CMPZV, EXTV, EXTZV"

                                          ;HERE WITH SIZE OPERAND IN D, POSITION IN Q

                                          34E:
                                          FF:
                                          CMPV:
                                          EXTV:    ;--------------------------------;CALL SITE FOR INVOKING ASPC SUBR
                                                   SC_Q,                            ;MOVE POSITION (LOW BITS) TO SC
                                                   R[R15]_Q,                        ;SAVE POSITION IN R15 FOR FFX
                                                   Q_Q.RIGHT,SI/ASHR,               ;GET POSITION/2
                                                   C[R.SD&SS,                       ;INIT SS FOR BRANCHES
U 034E, 0001,203C,C0B7,FAF8,0096,6054              EALU_K[.FFFF],CLK.UBCC           ;SET EALU N=1, Z=0

                                          =10****0;---------------------------------;CALL SITE FOR VSRC SPECIFIER EVALUATION
                                                   Q_Q.RIGHT2,SI/ASHR,              ;GET POSITION/8
                                                   FE_SC,                           ;MOVE LOW BITS OF POSITION TO FE
                                                   SC_D.OXT[BYTE]-K[.1],            ;GET SIZE -1 TO SC
U 0054, 001B,8001,0470,F800,0182,09AC              CALL,J/EXTV.2                    ;GO CALCULATE BASE

                                          =11****0;---------------------------------;RETURN HERE WITH BASE ADDRESS IN D
                                                   Q_Q+D,                           ;(POSITION/8)+BASE TO Q
                                                   D_D.LEFT,                        ;BASE *2
                                                   FE_SC-K[.1F],                    ;GET S-32 IN FE
                                                   SC_FE,                           ;LOW BITS OF POSITION TO SC
U 0074, 051D,3414,8DC0,F800,0185,A654              SC?,J/EXTV.3                     ;TEST SIZE .LEQU. 32

                                                   ;--------------------------------;RETURN HERE WITH REGISTER
                                                   Q_D,                             ;SAVE LOW REGISTER IN Q A MOMENT
                                                   D_Q.RIGHT2,                      ;D IS POSITION/32
                                                   FE_SC-K[.1F],                    ;GET S-32 TO FE
                                                   SC_FE,                           ; AND POSITION TO SC
U 0075, 0881,343C,8DE0,F800,0185,A614              SC?

                                          =100     ;--------------------------------;SC .EQL. 0 (SIZE =1)
                                                   SC_SC+FE,                        ;GET P+S-32
                                                   D_Q,                             ;GET LOW REGISTER WITH FIELD
                                                   Q_R(PRN+1),                      ;GET HIGH REGISTER
U 0614, 0C00,0D3C,01C0,F860,0080,8395              D.NE.0?,J/EXTV.5                 ;MAKE SURE POSITION WAS LESS THAN 32

                                                   ;--------------------------------;SC .LSS. 0 (SIZE =0)
                                                   D_0,Q_0,                         ;FIELD OF SIZE 0 IS ZERO
                                                   SC_FE,                           ;GET SIZE -32 FOR FFX USE
U 0615, 0F00,003F,01F8,F800,0081,0482              SUB/SPEC,J/EXTV.8                ;FIND OUT WHAT TO USE IT FOR

                                                   ;--------------------------------;0 .LSS. SC .LSS. 32 (SIZE VALID)
                                                   SC_SC+FE,                        ;GET P+S-32
                                                   D_Q,                             ;GET LOW REGISTER WITH FIELD
                                                   Q_R(PRN+1),                      ;GET HIGH REGISTER
U 0616, 0C00,0D3C,01C0,F860,0080,8395              D.NE.0?,J/EXTV.5                 ;MAKE SURE POSITION WAS LESS THAN 32

                                                   ;--------------------------------;SC .GEQ. 32
U 0617, 0000,003C,0180,F800,0000,0106              J/RSVOPR
                                          =;END OF SC TEST
```

```
                                         ;14432  ;HERE TO GET BASE OF FIELD.
                                         ;14433  ; RETURN ABOVE, WITH POSITION/8 IN Q, BASE ADDRESS IN D
                                         ;14434
                                         ;14435          ;-------------------------------;
                                         ;14436  EXTV.2: D_Q,                             ;POSITION /8 NOW IN D
                                         ;14437          LAB_R(SP1),                     ;LOAD LATCHES WITH SPECIFIER REGISTER
                                         ;14438          Q_IB.DATA,                      ;ASK IB FOR SPECIFIER DATA
                                         ;14439          CLR.IB.COND,                    ;CLEAR IT
                                         ;14440          PC_PC+N,                        ;STEP PC OVER IT
U 09AC, FC00,003F,01F0,F847,0000,0400    ;14441          SUB/SPEC,J/ASPC.B               ;GO EVALUATE BASE SPECIFIER ADDRESS
                                         ;14442
                                         ;14443  ;HERE FOR FIELD INSTRUCTIONS WHEN THE FIELD IS
                                         ;14444  ; IN MEMORY, AS OPPOSED TO A REGISTER.
                                         ;14445
                                         ;14446  =100    ;-------------------------------;SC .EQL. 0 (SIZE =1)
                                         ;14447  EXTV.3: VA_Q.ANDNOT.K[.3],              ;LONGWORD ADDRESS OF FIELD
                                         ;14448          D_D.LEFT2,                      ;BASE NOW *8 IN D
U 0654, 0119,2024,0D80,F800,0200,09AD    ;14449          J/EXTV.4
                                         ;14450
                                         ;14451          ;-------------------------------;SC .LSS. 0 (SIZE =0)
                                         ;14452          D_0,Q_0,                        ;ANY FIELD OF SIZE 0 IS ZERO
                                         ;14453          SC_FE,                          ;GET SIZE -32 FOR FFX
U 0655, 0F00,003F,01F8,F800,0081,0482    ;14454          SUB/SPEC,J/EXTV.8               ;GO FIND OUT WHAT TO DO WITH IT
                                         ;14455
                                         ;14456          ;-------------------------------;1 .LEQ. SC .LEQ. 31 (SIZE .LEQ. 32)
                                         ;14457          VA_Q.ANDNOT.K[.3],              ;LONGWORD ADDRESS OF FIELD
                                         ;14458          D_D.LEFT2,                      ;BASE NOW *8 IN D
U 0656, 0119,2024,0D80,F800,0200,09AD    ;14459          J/EXTV.4
                                         ;14460
                                         ;14461          ;-------------------------------;SC .GTR. 31 (SIZE .GTR. 32)
U 0657, 0000,003C,01[illegible]0,F800,0000,0106    ;14462          J/RSVOPR                        ;TAKE RESERVED OPERAND FAULT
                                         ;14463
                                         ;14464  =;END OF SC TEST
                                         ;14465
                                         ;14466          ;-------------------------------;
U 09AD, 0019,0014,1DC0,F800,0000,09AE    ;14467  EXTV.4: Q_D+K[SC]                       ;(BASE *8) + POSITION
                                         ;14468
                                         ;14469          ;-------------------------------;
                                         ;14470          SC_Q.AND.K[.1F],                ;GET P (POSITION IN LONGWORD) TO SC
U 09AE, 00[illegible],2034,8D80,4000,0082,09B0    ;14471          D[LONG]_CACHE                   ;GET LONGWORD CONTAINING FIELD
```

```
                                     ;AT THIS POINT, WE HAVE CALCULATED P, THE POSITION OF THE FIELD
                                     ;WITHIN AN ALIGNED LONGWORD, AND HAVE GOTTEN THAT LONGWORD CONTAINING THE
                                     ;LOW-ORDER BIT OF THE FIELD INTO D.  WE MUST DETERMINE WHETHER THE FIELD
                                     ;EXTENDS INTO A SECOND LONGWORD, AND SHIFT THE FIELD INTO PLACE.

                                               ;--------------------------------;
                                               SC_SC+FE,                        ;SC HAS P+S -32
                                               CLR.UBCC,                        ;IT MIGHT BE NEGATIVE. CHECK
                                               Q_D,                             ;COPY FIELD TO Q
U 09B0, 0000,003C,01E0,F803,0090,8395          VA_VA+4                          ;PREPARE TO GET NEXT LONGWORD IF NEEDED

                                     ;NOW SET UP D, Q AND SC FOR THIS SHIFT:
                                     ;               D               Q
                                     ;         *---------------*---------------*
                                     ;         *   ! <S> ! <P> *   ! <S> ! <P> *
                                     ;         *---------------*---------------*
                                     ;             *               *
                                     ;             *---------------*

                                     =101      ;--------------------------------;D.EQL.0 (REGISTER MODE, POS.LSS.32)
                                     EXTV.5:   SC_0-K[SC],                      ;GET 32- (P+S)
U 0395, 001B,1200,1D80,F800,0082,02E2          EALU?,J/EXTV.6                   ;TEST FOR P+S .GTR. 32
                                                                                ;IF REGISTER, EALU N=1, Z=0.

                                               ;--------------------------------;D.NEQ.0 (REGISTER MODE, POS.GEQ.32)
U 0397, 0000,003C,0180,F800,0000,0106          J/RSVOPR

                                     =001*     ;--------------------------------;EALU N&Z =0 (P+S .GTR. 32)
                                     EXTV.6:   D[LONG]_CACHE,                   ;GET SECOND PART  OF FIELD
                                               SC_SC+K[.20],                    ;GET 64-P+S
U 02E2, 0000,003C,7580,4000,0084,82FA          J/EXTV.7                         ;GO SHIFT

                                               ;--------------------------------;EALU N=0, Z=1 (P+S .EQL. 32)
                                               Q_0,                             ;READY TO ZERO EXTEND
                                               SC_FE,                           ;FIELD IS LEFT ADJUSTED IN D ALREADY
U 02E6, 0000,003F,01F8,F800,0081,0482          SUB/SPEC,J/EXTV.8                ;WHAT TO DO WITH IT?

                                               ;--------------------------------;EALU N=1 (P+S .LSS. 32)
                                     EXTV.7:   D_DAL.SC,                        ;MOVE SELECTED FIELD TO D<31:(32-S)>
                                               SC_FE,                           ;GET S-32 IN SC
                                               Q_0,                             ;PREPARE T[illegible]ERO EXTEND
U 02EA, 0D00,003F,01F8,F800,0081,0482          SUB/SPEC,J/EXTV.8                ;NOW, WHO W[illegible]TED THIS?

                                     =;END OF EALU N+Z TEST
```

```
                                          ;14516  ;AT THIS POINT, THE DESIRED FIELD HAS BEEN POSITIONED IN D, SO THAT ITS
                                          ;14517  ; MOST SIGNIFICANT BIT IS IN D31, AND SC CONTAINS THE FIELD SIZE -32.
                                          ;14518  ;               Q               D
                                          ;14519  ;        *---------------*---------------*
                                          ;14520  ;        *       0       * < S > !       *
                                          ;14521  ;        *---------------*---------------*
                                          ;14522  ;                *               *
                                          ;14523  ;                *---------------*
                                          ;14524
                                          ;14525  482:    ;------------------------------;EXTV OR CMPV
U 0482, 0000,0D3C,0180,F800,0000,048E     ;14526  EXTV.8: D31?,J/EXTV.9                   ;TEST SIGN OF FIELD
                                          ;14527
                                          ;14528  ;HERE WE EXTEND THE FIELD TO A LONGWORD, AND EVALUATE THE FINAL SPECIFIER
                                          ;14529  ; IF THIS IS EXTV OR EXTZV, THE FINAL SPECIFIER WILL BE 'WRITE'' TYPE,
                                          ;14530  ; AND THE CODE AT B-FORK WILL TRANSFER TO IRD AFTER STORING THE RESULT.
                                          ;14531  ; IF THIS IS CMPV OR CMPZV, THE FINAL SPECIFIER IS 'READ'' TYPE,
                                          ;14532  ; AND B-FORK WILL GO ON TO EXECUTION AT CMP.
                                          ;14533
                                          ;14534  48E:    ;------------------------------;D31=0. FIELD IS POSITIVE (OR ZERO-EXT)
                                          ;14535  EXTV.9: D_DAL.SC,                       ;SIGN- OR ZERO- EXTEND THE FIELD
U 048E, FD00,003F,01F0,F847,0000,0200     ;14536          B.FORK                          ;GO EVALUATE THE FOURTH SPECIFIER
                                          ;14537
                                          ;14538  48F:    ;------------------------------;D31=1. FIELD IS NEGATIVE
U 048F, 0001,2028,01C0,F800,0000,048E     ;14539          Q_NOT.Q,J/EXTV.9                ;SIGN-EXTENSION REQUIRES ONES
```

```
                                          ;HERE FOR FFS, FFC WITH THE SELECTED FIELD LEFT-ALIGNED IN D, AND Q ZERO.
                                          ; SC HAS S-32, FOR SHIFTING THE FIELD INTO D, RIGHT ALIGNED.
                                          ; R15 CONTAINS THE ORIGINAL POSITION.
                                          ; IF THIS IS FFC, WE FIRST COMPLEMENT D TO CONVERT ''CLEAR'' TO ''SET''

                                          481:     ;--------------------------------;
U 0481, 0801,0028,0180,F800,0000,0480     FFC.1:   D_NOT.D                           ;SEARCH FOR ZEROS

                                          480:     ;--------------------------------;
                                          FFS.1:   D_DAL.SC,                         ;RIGHT-ALIGN THE FIELD IN D
                                                   SC_SC+K[.20],                     ;RECOVER FIELD SIZE
U 0480, 0D18,0038,7550,F800,00F4,89B1              ALU_K[.20],LONG,CCK/INST.DEP      ;GUESS WE WILL FIND A BIT

                                                   ;--------------------------------;
                                                   Q_0-D,                            ;NEGATE D TO Q
U 09B1, 001F,2D00,01C0,F800,0000,04E5              D.NE.0?                           ;ARE ANY SET (FFS) OR CLEAR (FFC)?

                                          =101     ;--------------------------------;D.EQL.0
                                                   ALU_D,SET.CC(INST),               ;MAKE ALU ZERO, SET PSL<Z>
U 04E5, 0001,C03C,0180,F800,0070,09B4              J/FFS.3                           ;GO ADD SIZE TO OLD POSITION

                                                   ;--------------------------------;D.NEQ.0
                                                   D_D.AND.Q,                        ;LEAVE ONLY LEAST SIGNIFICANT ONE SET
U 04E7, 081D,0034,8D80,F800,0084,69B4              SC_K[.1F]                         ;SETUP 31 FOR CALCULATING BIT POSITION

                                                   ;--------------------------------;
                                          FFS.3:   SC_SC-SHF.VAL,                    ;SC GETS BIT POSITION OF FIRST ONE
                                                                                     ; (SHF.VAL =0 IF D=0)
U 09B4, 0000,003C,01C0,FA78,008C,A9B5              Q_R[R15]                          ;GET STARTING POSITION

                                                   ;--------------------------------;
                                                   D_Q+K[SC],                        ;GET POSITION TO RETURN
U 09B5, F819,2017,1DF0,F847,0000,0300              WRITE.DEST,J/WRD                  ;GIVE IT BACK
```

```
                                        .TOC    "       Field instructions     : INSV"

                                        ;THE VALUE TO BE INSERTED IS IN Q, AND THE FIELD POSITION IS IN D
                                        ;TEMPS USED BY THIS ROUTINE ARE:
                                        ;       ID[T0] = SOURCE, THE VALUE TO BE INSERTED
                                        ;       RC[T1] = POSITION/8
                                        ;       RC[T2] = ADDRESS OF LONGWORD CONTAINING LOW BIT(S) OF FIELD
                                        ;       RC[T3] = THE MEMORY LONGWORD ADDRESSES BY T2, MASKED DOWN
                                        ;                TO ONLY THE BITS TO BE PRESERVED.
                                        ;       RC[T4] = MASKS

                                        348:    ;--------------------------------;
                                        INSV:   SC_D,                             ;LOW BITS OF POSITION TO SC
                                                Q_D.RIGHT2,SI/ASHR,               ;POSITION/4 IN Q
                                                D_Q,                              ;PREPARE NEW VALUE FOR SAVING IN ID TO
                                                CLR.SD&SS,                        ;INIT FLAGS FOR EASIER BRANCHING
U 0348, 0C81,003D,00C7,F80C,0082,010A           CALL,J/INSV.1                     ;GO GET SIZE AND BASE OPERANDS

                                        368:    ;--------------------------------;RETURN HERE WITH BASE IN D IF MEMORY
                                                LC_RC[T1],                        ;GET POSITION/8 READY
                                                Q_D.LEFT3,                        ;ALSO BASE *8
                                                FE_SC-K[.1F],                     ;GET S-32 TO FE
                                                SC_FE,                            ;LOW BITS OF POSITION TO SC AGAIN
U 0368, 00A1,143C,8DC0,F908,0185,A684           SC?,J/INSV.4                      ;VALIDATE SIZE OPERAND

                                        369:    ;--------------------------------;RETURN HERE IF BASE SELECTS REGISTER
                                                D_Q.RIGHT2,SI/ZERO,Q_0,           ;GET P/32 IN D
                                                FE_SC-K[.1F],                     ;GET S-32 IN FE
                                                SC_FE,                            ;POSITION TO SC
U 0369, 0881,343C,8DF8,F800,0185,A674           SC?                               ;TEST SIZE-1

                                        =100    ;--------------------------------;SC=0 (SIZE=1)
                                                Q_ID[T0],                         ;GET DATA TO BE INSERTED
                                                RC[T4]_0+MASK+1,                  ;BEGIN MASK GENERATION
                                                FE_SC+FE,CLK.UBCC,                ;CALCULATE P+S-32
U 0674, 0003,0D10,C1F0,2DA0,0110,84FC           D.NE.0?,J/INSV.2                  ;IS POSITION LEGAL?

                                                ;--------------------------------;SC.LSS.0 (SIZE =0)
                                                CLR.IB.OPC,                       ;GO ON TO NEXT INSTR
U 0675, C000,003C,0180,F804,4000,0062           PC_PC+1,J/IRD

                                                ;--------------------------------;SC.GTR.0
                                                Q_ID[T0],                         ;GET DATA TO BE INSERTED
                                                RC[T4]_0+MASK+1,                  ;BEGIN MASK GENERATION
                                                FE_SC+FE,CLK.UBCC,                ;CALCULATE P+S-32
U 0676, 0003,0D10,C1F0,2DA0,0110,84FC           D.NE.0?,J/INSV.2                  ;IS POSITION LEGAL?

                                                ;--------------------------------;SC.GEQ.32
U 0677, 0000,003C,0180,F800,0000,0106           J/RSVOPR
```

```
                                             ;HERE TO INSERT A FIELD INTO A REGISTER

                                             =0*        ;-------------------------------;D.EQL.0 (POSITION .LSSU. 32)
                                             INSV.2:  D_Q,                              ;PREPARE TO ROTATE INSERT DATA
U 04FC, 0C00,123C,0180,F800,0000,0196                 EALU.N?,J/INSV.2A                 ;IS FIELD ALL IN ONE REGISTER?

                                                        ;-------------------------------;D.NEQ.0 (POSITION .GEQU. 32)
U 04FE, 0000,003C,0180,F800,0000,0106                 J/RSVOPR

                                             =0'1*      ;-------------------------------;EALU N=0 (P+S .GEQ.32)
                                             INSV.2A:D_DAL.SC,                          ;ALIGN THE INSERTION
                                                      SC_FE,                            ;P+S-32
                                                      Q_[A.ANDNOT.RC[T4],               ;SAVE NON-FIELD BITS OF REGISTER
U 0196, 0D10,0024,01C0,F920,0081,09C4                 J7INSV.3

                                             ; ***********************************************
                                             ; * Patch no. 065, PCS 0196 trapped to WCS 118B *
                                             ; ***********************************************

                                                        ;-------------------------------;EALU N=1 (P+S .LSS. 32)
                                                      D_DAL.SC,                         ;ALIGN THE INSERTION
                                                      SC_FE,                            ;GET P+S-32
U C19E, 0D10,0038,01C0,F920,0081,09B8                 Q_RC[T4]                          ;GET PARTIAL MASK

                                                        ;-------------------------------;
U 09B8, 0001,2008,01C0,F800,0000,09B9                 Q_Q-MASK-1                        ;PREPARE MASK OF FIELD

                                                        ;-------------------------------;
U 09B9, 081D,0034,0180,F800,0000,09BC                 D_D.AND.Q                         ;STRIP OFF ALL BUT INSERTION

                                                        ;-------------------------------;
U 09BC, 001C,0024,01C0,F858,0000,09BE                 Q_R(PRN).ANDNOT.Q                 ;CLEAR FIELD FROM REGISTER

                                                        ;-------------------------------;
                                                      R(PRN)_D.OR.Q,                    ;COMBINE FIELD WITH REST OF REGISTER
                                                      CLR.IB.OPC,                       ;GO TO NEXT INSTR
U 09BE, C01D,0030,0180,F8DC,4000,0062                 PC_PC+1,J/IRD                     ;
```

```
                                        ;14659  ;HERE FOR REGISTER INSERTION WHEN P+S IS GREATER THAN 32
                                        ;14660
                                        ;14661           ;-------------------------------;
U 09C4, 0811,0034,0180,F800,0000,09C5   ;14662  INSV.3:  D_D.AND.LC                        ;LOW PART OF I[illegible]ION INTO DATA
                                        ;14663
                                        ;14664           ;-------------------------------;
                                        ;14665           R(PRN)_D.OR.Q,                    ;COMBINE OLD DATA WITH INSERT
U 09C5, 001D,0030,C1F0,2CD8,0000,09C8   ;14666           Q_ID[TO]                          ;GET BACK REST OF INSERT
                                        ;14667
                                        ;14668           ;-------------------------------;
                                        ;14669           Q_0+MASK+1,                       ;MASK FOR HIGH PART OF INSERT
U 09C8, 0D03,0010,01C0,F800,0000,09C9   ;14670           D_DAL.SC                          ;GET HIGH PART OF INSERT
                                        ;14671
                                        ;14672  FI.PA.65:
                                        ;14673           ;-------------------------------;
U 09C9, 081D,0024,0180,F800,0000,09CA   ;14674           D_D.ANDNOT.Q                      ;STRIP INSERT DATA
                                        ;14675
                                        ;14676           ;-------------------------------;
U 09CA, 001C,0034,01C0,F860,0000,09CC   ;14677           Q_R(PRN+1).AND.Q                  ;CLEAR INSERTION PART OF REGISTER
                                        ;14678
                                        ;14679           ;-------------------------------;
                                        ;14680           R(PRN+1)_D.OR.Q,                  ;COMBINE INTO HIGH REGISTER
                                        ;14681           CLR.IB.OPC,PC_PC+1,               ;GO TO NEXT INSTR
U 09CC, C01D,0030,0180,F8E4,4000,0062   ;14682           J/IRD
```

```
                                        ;14683  ;HERE IN INSV TO GET THE SIZE AND BASE OPERANDS
                                        ;14684
                                        ;14685  =0**1*  ;-------------------------------;CALL SITE FOR SIZE SPECIFIER EVALUATION
                                        ;14686  INSV.1: ID[T0]_D,                       ;SAVE NEW VALUE FOR FIELD
                                        ;14687          RC[T1]_Q.RIGHT,SI/ASHR,         ;POSITION/8 NOW IN RC T1
                                        ;14688          D_Q.RIGHT,                      ; AND D
U 010A, 0841,203D,C080,3D88,0000,037E   ;14689          CALL,J/SPEC                     ;GO GET SIZE
                                        ;14690
                                        ;14691          ;-------------------------------;RETURN HERE WITH SIZE IN D
                                        ;14692          FE_SC,                          ;MOVE POSITION OUT OF THE WAY
                                        ;14693          SC_D.OXT[BYTE]-K[.1],           ;GET SIZE-1 TO SC
                                        ;14694          D_Q,                            ;LEAVE POS/8 IN Q ON RETURN
                                        ;14695          LAB_R(SP1),                     ;PREPARE TO EVALUATE BASE SPECIFIER
                                        ;14696          Q_IB.DATA,
                                        ;14697          CLR.IB.COND,
                                        ;14698          PC_PC+N,
U 011A, FC1B,8003,05F0,F847,0182,0400   ;14699          SUB/SPEC,J/ASPC.B               ;CALCULATE BASE ADDRESS
                                        ;14700
                                        ;14701  ;HERE FOR INSV ON A MEMORY FIELD
                                        ;14702
                                        ;14703  =100    ;-------------------------------;SC .EQL. 0 (SIZE =1)
                                        ;14704  INSV.4: SC_Q+K[SC],                     ;(BASE*8) + POS TO SC
                                        ;14705          LC_RC[T1],                      ;LOAD LATCH WITH POSITION/8
U 0684, 0019,2014,1D80,F908,0082,09CD   ;14706          J/INSV.5
                                        ;14707
                                        ;14708          ;-------------------------------;SC .LSS. 0 (SIZE =0)
                                        ;14709          CLR.IB.OPC,                     ;DO NOTHING
U 0685, C000,003C,0180,F804,4000,0062   ;14710          PC_PC+1,J/IRD
                                        ;14711
                                        ;14712          ;-------------------------------;0 .LSS. SC .LSS. 32 (SIZE VALID)
                                        ;14713          SC_Q+K[SC],                     ;SC GETS (BASE*8) + POS
                                        ;14714          LC_RC[T1],                      ;GET POSITION OFFSET INTO LATCH
U 0686, 0019,2014,1D80,F908,0082,09CD   ;14715          J/INSV.5
                                        ;14716
                                        ;14717          ;-------------------------------;SC .GEQ. 32 (SIZE .GTR. 32)
U 0687, 0000,003C,0180,F800,0000,0106   ;14718          J/RSVOPR
                                        ;14719
                                        ;14720  =;END OF SC TEST
```

```
                                              ;14721  ;HERE ON INSV OF A MEMORY OPERAND, HAVING VERIFIED THE SIZE
                                              ;14722  ; AND CALCULATED P, THE POSITION WITHIN AN ALIGNED LONGWORD
                                              ;14723
                                              ;14724          ;---------------------------------------;
                                              ;14725  INSV.5: SC_SC.ANDNOT.K[.FFE0],              ;SC IS NOW POSITICN IN LONGWORD
U 09CD, 0811,0014,A180,F800,0084,49CE         ;14726          D_D+LC                                ;D IS BYTE ADDRESS OF FIELD
                                              ;14727
                                              ;14728          ;---------------------------------------;
                                              ;14729          VA_D.ANDNOT.K[.3],                    ;CALCULATE LONGWORD ADDRESS OF FIELD
                                              ;14730          RC[T2]_ALU,                           ;SAVE IT FOR WRITING BACK
                                              ;14731          FE_SC+FE,                             ;P+S-32
U 09CE, 0019,0024,0D80,F990,0310,89D0         ;14732          CLR.UBCC                              ;WATCH FOR P+S .GTR. 32
                                              ;14733
                                              ;14734          ;---------------------------------------;
                                              ;14735          Q_0+MASK+1,                           ;ZEROS TO RIGHT OF FIELD
                                              ;14736          RC[T4]_ALU,                           ;KEEP HANDY
                                              ;14737          SC_FE,                                ;GET P+S-32 TO SC
                                              ;14738          FE_SC-K[.20],                         ;P-32 TO FE
                                              ;14739          D[LONG]_CACHE.WCHK,                   ;GET FIELD
U 09D0, 0003,1210,75C0,51A0,0185,A2F6         ;14740          EALU.N?                               ;DOES IT CROSS LONGWORD BOUNDARY?
                                              ;14741
                                              ;14742  ; ***********************************************
                                              ;14743  ; * Patch no. 036, PCS 09D0 trapped to WCS 1170 *
                                              ;14744  ; ***********************************************
                                              ;14745
                                              ;14746  =011*   ;---------------------------------------,EALU N=0, FIELD LIES ACROSS BOUNDARY
                                              ;14747          RC[T3]_D.ANDNOT.Q,                    ;SAVE LOW PART OF MEMORY DATA
                                              ;14748          Q_ID[T0],                             ;GET DATA TO BE INSERTED
                                              ;14749          SC_FE,                                ;GET P-32 FOR ALIGNING INSERT DATA
                                              ;14750          FE_SC,                                ;SAVE P+S-32 (WHICH IS POSITIVE)
                                              ;14751          VA_VA+4,                              ;ADDR OF SECOND LONGWORD
U 02F6, 001D,0024,C1F0,2D9B,0181,09D5         ;14752          J/INSV.6
                                              ;14753
                                              ;14754          ;---------------------------------------;EALU N=1, FIELD IS IN ONE LONGWORD
                                              ;14755          RC[T4]_Q-MASK-1,                      ;MASK FOR BITS OF MEMORY TO DISCARD
                                              ;14756          SC_FE,                                ;GET P-32 FOR SHIFT
U 02FE, 0001,2008,C1F0,2DA0,0081,09D1         ;14757          Q_ID[T0]                              ;GET DATA TO INSERT
                                              ;14758
                                              ;14759  =;END OF EALU N TEST
                                              ;14760
                                              ;14761  FI.PA.36:
                                              ;14762          ;---------------------------------------;
                                              ;14763          Q_D.ANDNOT.RC[T4],                    ;CLEAR THE FIELD OF MEMORY WORD
U 09D1, 0D11,0024,01C0,F920,C000,09D4         ;14764          D_DAL.SC                              ;ALIGN THE INSERT DATA
                                              ;14765
                                              ;14766          ;---------------------------------------;
                                              ;14767          D_D.AND.LC,                           ;DROP JUNK NOT TO BE INSERTED
U 09D4, 0811,0034,0180,F800,0000,09E1         ;14768          J/INSV.7                              ;GO COMBINE AND STORE
```

```
                                           ;HERE WHEN THE FIELD LIES ACROSS A LONGWORD BOUNDARY


                                                     ;--------------------------------;
                                           INSV.6:  D_DAL.SC,                          ;ALIGN THE INSERT DATA BY POSITION
                                                    FE_SC+K[.20],                      ;RESTORE P TO FE
U 09D5, 0D00,003C,7530,F800,0185,89D6               SC_FE                              ;AND P+S-32 TO SC

                                                     ;--------------------------------;
                                                    Q_D.AND.RC[T4],                    ;GET DESIRED PART OF THE INSERT
U 09D6, 0011,0034,01C0,5120,0000,09D8               D[LONG]_CACHE.WCHK                 ; AND THE OTHER MEMORY DATA

                                                     ;--------------------------------;
                                                    VA_RC[T2],                         ;RELOAD ADDR OF FIRST LONGWORD
                                                    Q_D,                               ;HOLD NEXT MEMORY DATA AT SIDE
U 09D8, 0C10,0038,01E0,F910,0200,09D9               D_Q                                ;GET THE INSERT FIELD INTO D

                                                     ;--------------------------------;
U 09D9, 0811,0030,0180,F918,0000,09DC               D_D.OR.RC[T3]                      ;COMBINE NEW INSERT WITH OLD MEMORY DATA

                                                     ;--------------------------------;
                                                    CACHE_D[LONG],                     ;WRITE BACK LOW PART OF FIELD
                                                    RC[T4]_0+MASK+1,                   ;GET MASK FOR MEM SECOND PART
U 09DC, 0003,0010,0180,31A0,0081,09DD               SC_FE                              ;P TO SC

                                                     ;--------------------------------;
                                                    D_Q.AND.RC[T4],                    ;STRIP LOW BITS FROM SECOND PART
                                                    VA_VA+4,                           ;GET ADDR OF HIGH PART AGAIN
U 09DD, 0811,2034,01F0,2D23,0000,09DE               Q_ID[T0]                           ;GET INSERT DATA AGAIN

                                                     ;--------------------------------;
                                                    Q_D,                               ;SAVE MEM PART2
U 09DE, 0D00,003C,01E0,F800,0000,09E0               D_DAL.SC                           ;ALIGN THE INSERT

                                                     ;--------------------------------;
U 09E0, 0811,0024,0180,F800,0000,09E1               D_D.ANDNOT.LC                      ;DISCARD JUNK FROM INSERT

                                                     ;--------------------------------;
                                           INSV.7:  D_D.OR.Q,                          ;COMBINE INSERT WITH MEMORY DATA
U 09E1, 081D,0030,0180,F800,0000,03FD               J7STOR.L                           ;PUT IT BACK IN MEMORY

                                           .LIST             ;Re-enable full listing
```

```
                                                ;14810  .TOC     "CHAR.MIC"
                                                ;14811  .TOC     "Revision 1.3"
                                                ;14812  ;         P. R. Guilbault
                                                ;14813
;14814  .NOBIN
;14815  .TOC    ''      Revision History''
;14816
;14817  ; 01    Change macro names that deal with conditions codes.
;14818  ;       Comment patch no. 095 MOVC5 optimization.
;14819  ; 00    Create this file by merging MACCOM.MIC, MOV.MIC, SKP.MIC, SPAN.MIC, CMP.MIC, MATCH.MIC, and MTC.MIC
;14820  ;       Start of history
;14821
                                                ;14822  .BIN
                                                ;14823  .NOLIST          ;Disable listing of PCS code for quickie assemblies
```

```
                                        .TOC    "       Character string        : Utilities"

                                        ;ALGORITHM:
                                        ;               AS PART OF INITIALIZATION OF SOME STRING INSTRUCTIONS,
                                        ;               CLEAR PSL CONDITION CODES
                                        ;
                                        ;INPUTS:
                                        ;               CALLED WITH PSL IN Q
                                        ;
                                        ;OUTPUTS:
                                        ;               ID/PSL WRITTEN WITH COND CODES ALL 0
                                        ;
                                        ;RETURN:
                                        ;               ALWAYS RETURNS 40

                                                        ;--------------------------------;
                                        CLRPSLCC:
                                                        D_Q.ANDNOT.K[.F],               ;CLEAR PSL CC BITS
                                                        Q_D,                            ;PRESERVE D AS PASSED IN Q
U 09E4, 0819,2024,61E0,F800,0104,69E5                   FE_K[.F]                        ;FE IS USED AS FLAG FOR FPD
                                                                                        ;FE NOT 0 = EXCEPTION,
                                                                                        ;FE = 0 = INTERRUPT
                                                        ;--------------------------------;
                                                        ID[PSL]_D,                      ;PSL CC ALL CLEAR
                                                        D_Q,                            ;RESTORE D
U 09E5, 0C00,003E,3D80,3C00,0000,0040                   RETURN40                        ;

                                        ;ALGORITHM:
                                        ;               WHEN AN INTERRUPT OR EXCEPTION HAS BEEN NOTED BY A STRING INSTRUCTION,
                                        ;               THIS ROUTINE SAVES SOME VULNERABLE DATA IN R0<31:16> BEFORE
                                        ;               HANDLING THE INT/EXC SO THE INSTRUCTION CAN BE RESUMED.
                                        ;               JUMPS TO FPD.RTN IF AN EXCEPTION OR INTIO IF AN INTERRUPT PENDING.

                                        ;INPUTS:
                                        ;               STATE=8 BITS OF DATA TO BE SAVED(CONTENTS VARY WITH EACH INSTR)

                                        ;OUTPUTS:
                                        ;               R0<31:24>=PC DELTA, <23:16>=STATE, <15:00>=R0 AS PASSED

                                        =00             ;--------------------------------;
                                        FPDPACK:
                                                        SC_STATE,                       ;PREPARE TO SAVE STATE REG
                                                        CALL,J/BAKUP.PC,                ;BAKUP WILL RETURN PC DELTA
U 06D8, 0014,0039,01C0,F800,1480,0EB8                   Q_PC                            ;IF GIVEN THE CURRENT PC IN Q

                                        =10             ;--------------------------------;
                                                        D_D.SWAP,                       ;PC DELTA RETURNED IN D<7:0>
                                                                                        ;PUT IT IN D<31:24>
U 06DA, 0B18,0038,1DC0,F800,0000,09E6                   Q_SC                            ;STATE TO Q
                                        =
                                                        ;--------------------------------;
U 09E6, 0000,003C,7180,F800,0084,69E8                   SC_K[.FFF8]                     ;SC_-8 FOR RIGHT SHIFT 8
```

```
                                           FPDPACK1:
                                                   ;---------------------------------;
                                                   D_DAL.SC,                         ;STATE TO Q<31:24>,
                                                                                     ;PC DELTA TO D<23:16>
                                                   SC_FE,                            ;MOVE INT/EXC FLAG TO SC TO TEST IT
U 09E8, 0D18,0034,C1C0,FA00,0081,09E9              Q_R[R0].AND.K[.FFFF]              ;PRESERVE LOW WORD OF R0

                                                   ;---------------------------------;
U 09E9, 001D,0C30,0180,FA80,0000,04F3              R[R0]_D.OR.Q,BEN/MUL              ;R0=STATE,PC DELTA, R0<15:0>
                                           =011    ;---------------------------------;SC EQ 0?
U 04F3, 0000,0E3C,0180,F800,0000,0F8D              BEN/INTERRUPT,J/INTIO             ;SC NE 0, INTIO RTN WILL DECIDE

                                           ; ***********************************************
                                           ; * Patch no. 030, PCS 04F3 trapped to WCS 1162 *
                                           ; ***********************************************
                                                                                     ;IF INTERNAL OR EXTERNAL INTERRUPT
                                                   ;111------------------------------;
U 04F7, 0000,003C,0180,F800,0000,0EBB              J/FPD.RTN                         ;SC = 0. IT'S AN EXCEPTION

                                                   ;---------------------------------;

                                           ;ALGORITHM:
                                           ;       THIS IS A SEQUENCE OF INSTRUCTIONS THAT MERELY ZERO
                                           ;       REGISTERS AS REQUIRED BY THE SRM FOR TERMINATING ASSORTED
                                           ;       INSTRUCTIONS. THIS SEQUENCE CAN BE ENTERED AT ANY POINT
                                           ;       DEPENDING ON THE INDIVIDUAL INSTRUCTION'S REQUIREMENTS.

                                           ;INPUTS:
                                           ;       NONE

                                           ;OUTPUTS:
                                           ;       SPECIFIED REGISTERS ARE ZEROED
                                           ;       JUMPS TO IB.FILL WITH PSL<FPD> CLEARED

                                                   ;---------------------------------;
                                           R24503ZERO:
U 09EC, 0018,0038,1980,FA98,0000,09ED              R[R3]_K[ZERO]                     ;R3_0

                                                   ;---------------------------------;
                                           R0245ZERO:
U 09ED, 0018,0038,1980,FA80,0000,09EE              R[R0]_K[ZERO]                     ;R0_0

                                                   ;---------------------------------;
                                           R245ZERO:
U 09EE, 0018,0038,1980,FAA8,0000,09F0              R[R5]_K[ZERO]                     ;R5_0

                                                   ;---------------------------------;
                                           R24ZERO:
U 09F0, 0018,0038,1980,FAA0,0000,09F1              R[R4]_K[ZERO]                     ;R4_0

                                                   ;---------------------------------;
                                           R2ZERO: R[R2]_K[ZERO],                    ;R2_0
                                                   CLR.FPD,                          ;CLEAR FPD BIT
U 09F1, 0018,1238,1980,FA90,2000,05AE              BEN/EALU                          ;BRANCH ON REG WRITE FLAG
```

```
                                          ;14930  =1110
                                          ;14931  STRINGFINAL:
                                          ;14932          ;------------------------------;SIGN SRC
U 05AE, 2014,0038,0180,F801,4200,00AB     ;14933          PC&VA_PC,FLUSH.IB,J/IB.FILL     ;ALL DONE. BACK TO IB
                                          ;14934
                                          ;14935          ;1111--------------------------;
                                          ;14936          R(SC)_D.SET.CC(LONG),           ;WRITE CRC RESULT IN REG
U 05AF, 0001,003C,0180,F8E8,0070,05AE     ;14937          J/STRINGFINAL                   ;
                                          ;14938
                                          ;14939          ;------------------------------;
                                          ;14940
                                          ;14941  ;ALGORITHM:
                                          ;14942  ;       SHIFTS D + Q SO THAT PC CAN BE RESET FROM PC DELTA SAVED ACROSS INT/EXC.
                                          ;14943  ;       ALSO GETS STATE REG CONTENTS READY TO BE RESTORED/USED.
                                          ;14944  ;       CLEARS R0<31:16>
                                          ;14945
                                          ;14946  ;INPUTS:
                                          ;14947  ;       D+Q = SAVED R0, SC=-16
                                          ;14948
                                          ;14949  ;OUTPUTS:
                                          ;14950  ;       PC RESET, R0<31:16>=0, R0<15:0>UNMOLESTED,
                                          ;14951  ;       D<7:0> = STATE, ASSUMING SET BY FPDPACK,
                                          ;14952  ;       OTHERWISE WHATEVER WAS IN R0<31:24>
                                          ;14953
                                          ;14954  ;RETURN:
                                          ;14955  ;       ALWAYS RETURNS 100
                                          ;14956
                                          ;14957          ;------------------------------;
                                          ;14958  FPDUNPACK:
                                          ;14959          D_DAL.SC,                       ;D<7:0>=PC DELTA,<15:8>=STATE
U 09F2, 0D19,2034,C180,FA80,0000,09F4     ;14960          R[R0]_Q.AND.K[.FFFF]            ;RESTORE R0
                                          ;14961
                                          ;14962          ;------------------------------;
                                          ;14963          Q_D.AND.K[.FF],                 ;EXTRACT PC DELTA
                                          ;14964          FE_K[.FF],                      ;RESET FE SO LOOKS LIKE EXCEPTION
U 09F4, 0C19,0034,49C0,F800,0104,69F5     ;14965          D_Q                             ;D_R0 AS SAVED AT FPD TIME
                                          ;14966
                                          ;14967          ;------------------------------;
                                          ;14968          D_D.SWAP,                       ;D<7:0>=STATE
U 09F5, 0B15,2016,0180,F801,0200,0100     ;14969          PC_Q+PC,RETURN100               ;RESET PC
                                          ;14970          ;------------------------------;
```

```
.TOC    ""      Character string        : MOVC3, MOVC5"

;GENERAL OVERVIEW OF MOVC3 + MOVC5

;       REGISTER USAGE:
;R0     LENGTH OF STRING TO MOVE (BYTE-REVERSED, IN <31:16>)
;       STATE AND PC-DELTA (IN FLTG PT FORMAT IN <15:0>)
;R1     ADDR OF SOURCE
;R2     LENGTH OF FILL (-1) IN <15:0>; SIGN IS IN STATE<6>
;       IF POSITIVE, IMPLIES THERE'S FILL TO DO
;       IF NEGATIVE, IT'S COMPLEMENT OF EXCESS SRC,
;       I.E. DEST LENGTH - SRC LENGTH (-1) ; HENCE, NO FILL
;R3     DEST ADDR
;R5     ORIGINAL R0 DURING SRC MOVE.
;       4 BYTES OF FILL CHAR DURING FILL MOVE.
;STATE:
;       <1:0>   00 = LWD MOVE
;               01 = BYTE MOVE
;               10 = WORD MOVE
;       4       BACKWARDS(SRC MOVE IS IN BACKWARDS DIRECTION)
;       5       FILLING(VALID ONLY WHILE MOVING FILL)
;       6       NEED TO FILL(VALID ONLY WHILE MOVING SRC)
;       7       0 FOR MOVC3/5, 1 FOR MOVTC/TUC IN THOSE PLACES
;               WHERE THE TWO OPERATIONS SHARE CODE
;INITIALLY, IF SRC ADDR > DEST ADDR, THERE'S A POSSIBILITY OF OVERWRITING
;SRC BEFORE IT'S MOVED, SO R1 + R3 ADJUSTED FOR BACKWARDS MOVE
;NAMELY, R1_R1+R0, R3_R3+R0, R5_R0
;AFTER SRC IS MOVED FROM HIGHEST ADDR TO LOWEST, R1_R1+R5, R3_R3+R5 AND
;CODE JOINS FORWARD EXECUTION.
;AFTER SRC MOVE, IF R2 >= 0, R0_R2 + DONE
;IF R2 < 0, THERE'S FILL TO CONSIDER. R2_0, R5_ LWD OF FILL CHAR, SC NOT 0,
;R0_0-R2 (FILL COUNTER AS A POSITIVE NUMBER) + USES MAIN FORWARD CODE.
;WHICH BRINGS US TO FPD HANDLING:
;IN MAIN LOOP, SRC ADDR IN LB, DEST ADDR IN LA
;FOR COUNT, VARIES WITH OPERATION
;IN PARTICULAR, READ FAULT & WRITE FAULT - RC[T2].
;INTERRUPT - Q
```

```
                                         .TOC    "       Character string      : MOVC3/5 INITIALIZATION"

                                                 ;-------------------------------;
                                         44A:
                                         MOVC3:  RC[T0]_Q.0XT[WORD],             ;1ST ARG IS LENGTH
U 044A, 0003,603D,0180,F980,0000,047E            CALL,J7ASPC                     ;GET DEST ADDR

                                                 ;-------------------------------;
U 046A, 0001,003C,0180,FA98,0000,09F6    46A:    R[R3]_D                         ;SAVE DEST ADDR

                                                 ;-------------------------------;
U 09F6, 0001,203C,0180,FB80,0000,09F8            LC_RC[T0]&R1_ALU,ALU_Q          ;GET LENGTH + SAVE SRC ADDR

                                                 ;-------------------------------;
                                                 R[R2]_NOT.0,                    ;INITIALIZATION ONLY
U 09F8, 0003,0028,1980,FA90,1404,69F9            STATE_K[ZERO]

                                                 ;-------------------------------;
                                                 ALU_0-K[ZERO], SET.CC(LONG),    ;CHANGING C.C. ROM SAVES A CYCLE HERE
U 09F9, 001B,0000,1980,F800,0070,05B7            J/MOVC

                                                 ;-------------------------------;
                                         44E:
                                         MOVC5:  RC[T0]_Q.0XT[WORD],             ;SAVE SRC LENGTH
U 044E, 0003,603D,0180,F980,0000,037E            CALL,J7SPEC                     ;GET FILL BYTE

                                                 ;-------------------------------;
                                         45E:    RC[T1]_Q,                       ;SAVE SRC ADDR
                                                 Q_D,                            ;COPY FILL BYTE
                                                 D_D.SWAP,                       ;D<31:24> = FILL CHAR
                                                 SC_K[.FFF8],                    ;PREPARE FOR A DAL
U 045E, 0B01,203D,71E0,F988,0084,60A2            CALL, J/MOVCCMPC5               ;COLLECT DESTINATION LEN,ADDR

                                                 ;-------------------------------;
                                         0C5E:            ;RETURN FROM MOVCCMPC5 HERE WITH RC[T2] AND Q =DEST LEN,
                                         MOVC5SETUP:      ;R2<31:16>=2 FILL CHARS, RC[T1]=SRC ADDR, D=DEST ADDR,
                                                          ;RC[T0] = SRC LEN
                                                          ;**NOTE** - MOVTC & MOVTUC ENTER MOVC FLOWS HERE **

                                                 R[R3]_D,                        ;SAVE DEST ADDR
U 0C5E, 0C01,003C,0180,FA98,0000,09FA            D_Q                             ;GET DEST LENGTH

                                                 ;-------------------------------;
U 09FA, 0010,0038,01C0,F908,0000,09FC            Q_RC[T1]                        ;SRC ADDR

                                                 ;-------------------------------;
U 09FC, 0001,203C,0180,FB80,0000,09FD            LC_RC[T0]&R1_Q                  ;SAVE SRC ADDR IN R1, LATCH SRC LENGTH
```

```
                                                  ;---------------------------------;
                                                  R[R2]_D-LC-1, DT/WORD,            ;DEST LENGTH - SRC LENGTH - 1
                                                  SET.CC(WORD),                     ;SET CC'S ON WORD SUBTRACT A LA COMPARE
U 09FD, 0011,4008,3180,FA90,1474,69FE             STATE_K[.40]                      ;INITIALIZE STATE TO 'NEED TO FILL''

                                                  ;---------------------------------;
                                                  D_R[R3],                          ;LOAD DEST ADDR INTO D FOR COMPARE,
U 09FE, 0800,023C,0180,FA18,0000,05B5             BEN/ROR                           ;TEST IF DEST LEN > SRC LEN

                                         =101     ;---------------------------------;PSL <C>
U 05B5, 0000,003C,1980,F910,1404,65B7             LC_RC[T2], STATE_K[ZERO]          ;DEST<=SRC, NO FILLS, MIN = DESTLEN

                                         ;MOVC3 + MOVC5 CONVERGING POINT
                                         ;DEST ADDR IN D, SRC ADDR IN Q, MIN(SRCLEN,DESTLEN) IN LC

                                                  ;111------------------------------;
                                         MOVC:    Q_D-Q,                            ;DEST ADDR-SRC ADDR
                                                  C[K.UBCC,                         ;SEE IF FORWARDS OR BACKWARDS MOVE
                                                  SC_FE,
U 05B7, 001D,0900,01C0,F800,0091,06F2             IRT?                              ;TEST IF MOVC OR MOVTC/TUC
                                         =10      ;---------------------------------;IR<1>. BREAKOUT MOVC3/MOVC5
                                                                                    ;FROM MOVTC/MOVTUC USING IR1
                                                  D_K[.1F00].RIGHT,                 ;FAULT VECTORS FOR MOVC ARE F81,F82
                                                  LAB_R[R1],
                                                  SET.FPD,                          ;THIS IS AN INTERRUPTABLE INSTRUCTION
                                                  SC_SC-FE,                         ;CLEAR SC
U 06F2, 0858,0338,9180,FA08,2480,A59C             C31?, J/MOVCSETFPD

                                                  ;1*-------------------------------;
                                                  LAB_R[R1], D_K[.1F00].RIGHT,      ;GET MOVC FAULT VECTOR IN D,
                                                  Q_0, SS_0&SD_0, SET.FPD,          ;INIT FAULT FLAG & SET FPD,
U 06F3, 0858,1B38,91FF,FA08,2400,063C             IR0.C31?, J/MOVTCWHATDIR          ;BREAK OUT ON OP AND DIRECTION
```

```
                                         ;15087  =0*     ;-------------------------------;ALU <C>
                                         ;15088  MOVCSETFPD:
                                         ;15089          ID[FPDA]_D, SS_0&SD_0,         ;SD TELLS FAULTS FROM INTERRUPTS
                                         ;15090          Q_LC,                          ;SRC ADDR > DEST ADDR.
                                         ;15091          CLK.UBCC,
U 059C, 0010,0038,B5C7,3C00,0010,06B6    ;15092          J/MOVCFLP                      ;MOVE FORWARD
                                         ;15093
                                         ;15094          ;1*-----------------------------;
                                         ;15095          ID[FPDA]_D, SS_0&SD_0,         ;MAY BE BACKWARDS MOVE; IF SRC OUT
                                         ;15096          ALU_Q-LC,                      ;OF RANGE OF DEST, CAN MOVE FORWARD
                                         ;15097          LA_RA[3],                      ;LATCH DEST ADDR
                                         ;15098          CLK.UBCC,
U 059E, 0011,2000,B587,3C98,0010,0770    ;15099          J/MOVCBCKWDSMAYBE
                                         ;15100
                                         ;15101          ;-------------------------------;
                                         ;15102  ;       SUBROUTINE TO BUMP R1 & R3 BY AMOUNT IN Q AND SET BACKWARDS FLAG.
                                         ;15103  ;       ALSO SETS SC=R3<1:0>.  IF CALLED WITH BEN/C31 WITH C31 SET, DOES
                                         ;15104  ;       NOT RETURN BUT ENTERS FORWARD MOVE FLOWS.
                                         ;15105
                                         ;15106  =0*     ;-------------------------------;ALU <C>
                                         ;15107  MOVCRBUMP:
                                         ;15108          R[R3]_LA+Q,                    ;ADJUST DEST ADDR
                                         ;15109          SC_ALU,                        ;TO BE USED FOR OFFSET INDICATOR
                                         ;15110          STATE_STATE.OR.K[.10],         ;SET BACKWARDS BIT
U 05C4, 001C,0014,6580,FA98,1486,2A00    ;15111          J/MOVCRBUMP.1
                                         ;15112
                                         ;15113          ;1*-----------------------------;
                                         ;15114          D_R[R3],                       ;IT'LL BE FORWARD DIRECTION
                                         ;15115          Z?,                            ;ANY TO DO
U 05C6, 0800,013C,0180,FA18,0000,06C6    ;15116          J/MOVCFTST
                                         ;15117
                                         ;15118          ;-------------------------------
                                         ;15119  MOVCRBUMP.1:
                                         ;15120          R[R1]_Q+LB,                    ;ADJUST SRC ADDR
                                         ;15121          SC_SC.ANDNOT.K[.FFFC],         ;SAVE <1:0> OF DEST ADDR
U 0A00, 000D,2016,F180,FA88,0084,4002    ;15122          RETURN2
```

```
                                        ;15123  ;       SUBROUTINE TO GET A LENGTH/ADDRESS PAIR OF SPECIFIERS
                                        ;15124  ;       USED BY MOVC, CMPC, MOVTC, MOVTUC.
                                        ;15125
                                        ;15126          ;-------------------------------;
                                        ;15127  =010**1*                                ;HANDLE CALL,SPEC + CALL,ASPC SEQ
                                        ;15128  MOVCCMPC5:
                                        ;15129          D_DAL.SC,                       ;D<31:16> = 2BYTES OF FILL
                                        ;15130          A[U_0+K[.1],                    ;CLEAR OUT ALU CC
                                        ;15131          CLK.UBCC,                       ;HANDY FOR LOOSER BEN/IR LATER
U 00A2, 0D1B,0015,0580,F800,0010,037E   ;15132          CALL,J/SPEC                     ;WILL RETURN WITH D IN Q
                                        ;15133
                                        ;15134          ;-------------------------------;
                                        ;15135  =011**1*                                ;SPEC RETURNS 10
                                        ;15136          RC[T2]_D.OXT[WORD],             ;SAVE DEST LENGTH
                                        ;15137          D_Q,                            ;RESTORE 2 BYTES OF FILL CHAR
U 00B2, 0C03,403D,0180,F990,0000,047E   ;15138          CALL,J/ASPC                     ;THIS'LL PUT FILL BYTES BACK INTO Q
                                        ;15139
                                        ;15140          ;-------------------------------;
                                        ;15141  =111**1*
U 00F2, 0019,2024,C180,FA90,0000,0A01   ;15142          R[R2]_Q.ANDNOT.K[.FFFF]         ;SAVE 2 BYTES OF FILL IN R2<31:16>
                                        ;15143
                                        ;15144          ;-------------------------------;
                                        ;15145          Q_RC[T2],                       ;LOAD Q WITH (USUALLY) DEST LEN,
U 0A01, 0010,003A,01C0,F910,0000,0800   ;15146          RETURN[800]                     ;RETURN TO CALLER
```

```
                                        ;15147  .TOC    "       Character string        : MOVC3/5 MAIN LOOPS"
                                        ;15148
                                        ;15149  ;COUNT IN Q. SAVE IN RC[T2] ON EVERY ITERATION
                                        ;15150  ;ALU CC <Z> SET ON Q
                                        ;15151  ;SC = 0 = NOT FILL
                                        ;15152  ;R5 = 4 BYTES OF FILL CHAR
                                        ;15153
                                        ;15154  =110    ;-------------------------------;INTERRUPT
                                        ;15155  MOVCFLP:
                                        ;15156          D_R[R3],                        ;DEST ADDR
                                        ;15157          Z?,                             ;MORE TO DO?
U 06B6, 0800,013C,0180,FA18,0000,06C6   ;15158          J/MOVCFTST
                                        ;15159
                                        ;15160          ;111----------------------------;
                                        ;15161  MOVCINTF:
                                        ;15162          D_Q,                            ;GET LOOP COUNTER IN D FOR SWAP,
U 06B7, 0C00,003C,0180,F800,0000,0733   ;15163          J/MOVCPACKST                    ;GO PACK STUFF INTO R0
                                        ;15164
                                        ;15165  ;NEXT 2 STATES PROVIDE FOR A TIMELY SAVING OF 1 STATE IF PLACED AT
                                        ;15166  ;THIS LOCATION
                                        ;15167
                                        ;15168  =100    ;-------------------------------;
                                        ;15169  MOVCREADJUST:                           ;AFTER A BACKWARDS MOVE
U 06C4, 0000,003C,0180,FA08,0000,06C5   ;15170          LAB_R[R1]                       ;SRC ADDR TO LB
                                        ;15171
                                        ;15172          ;101----------------------------;
                                        ;15173          LA_RA[3],                       ;DEST ADDR TO LA
U 06C5, 0000,003D,0180,F898,0000,05C4   ;15174          CALL,J/MOVCRBUMP                ;WILL RETURN TO 111 OF CONSTRAINT
                                        ;15175
                                        ;15176          ;110----------------------------;PART OF A Z? ALSO
                                        ;15177  MOVCFTST:
                                        ;15178          D_Q-K[.3]-1, RC[T2]_ALU,        ;ASSUME 4 BYTES LEFT, SAVE NEW CT
                                        ;15179          C[K.UBCC,                       ;SET C31 IF 4 OR MORE BYTES LEFT
                                        ;15180          STATE_STATE.ANDNOT.K[.3],       ;ASSUME LONGWORD TRANSFER
                                        ;15181          BEN/MUL,                        ;BRANCH ON DEST OFFSET +  IF FILL
U 06C6, 0819,2C08,0D80,F990,1414,46E0   ;15182          J/MOVCOFFSET
                                        ;15183
                                        ;15184          ;111----------------------------;END OF MOVC FORWARD(DUE TO BEN) OR
                                        ;15185                                          ;FROM CALL AT MOVCREADJUST
                                        ;15186          D_NOT.R[R2], Q_NOT.R[R2],       ;GET -(FILL COUNT) IN D AND Q LOW
                                        ;15187          STATE_STATE.ANDNOT.K[.30],              ;CLEAR FILL + BACKWARDS BITS
                                        ;15188          BEN/STATE7-4,                   ;NEED TO FILL?
U 06C7, 0800,1628,79C0 F890,1404,47A3   ;15189          J/MOVCMAYBEFILL
```

```
                                          ;15190  =000       ;--------------------------------;BEN/MUL - ALL 8 WAYS
                                          ;15191  MOVCOFFSET:
                                          ;15192          LAB_R[R1],                        ;DEST ADDR = BYTE 0, NOT FILL
                                          ;15193          VA_[A,                            ;LOAD SRC ADDR
                                          ;15194          ID[T2]_D,                         ;SAVE UPDATE COUNT
                                          ;15195          C31?,                             ;IS THERE ROOM FOR THIS LWD?
U 06E0, 0000,033C,C980,3E08,0200,05C8     ;15196          J/MOVCFWC
                                          ;15197
                                          ;15198          ;001-----------------------------;DEST ADDR = BYTE 1, NOT FILL
                                          ;15199          LAB_R[R1],
                                          ;15200          VA_[A,
U 06E1, 0000,003C,0180,FA08,0200,05D9     ;15201          J/MOVCFB
                                          ;15202
                                          ;15203          ;010-----------------------------;DEST ADDR = BYTE 2, NOT FILL
                                          ;15204          LAB_R[R1],
                                          ;15205          VA_[A,
                                          ;15206          C31?,
U 06E2, 0000,033C,0180,FA08,0200,05DC     ;15207          J/MOVCWD
                                          ;15208
                                          ;15209          ;011-----------------------------;DEST ADDR = BYTE 3, NOT FILL
                                          ;15210          LAB_R[R1],
                                          ;15211          VA_[A,
U 06E3, 0000,003C,0180,FA08,0200,05D9     ;15212          J/MOVCFB
                                          ;15213
                                          ;15214          ;100-----------------------------;DEST ADDR = BYTE 0, FILL
                                          ;15215          LAB_R[R1],                        ;NECESSARY FOR FPD INTERFACE
                                          ;15216          Q_D,                              ;COPY DECREMENTED COUNTER
                                          ;15217          C31?,
U 06E4, 0000,033C,01E0,FA08,0000,05EC     ;15218          J/MOVCFILLMORE
                                          ;15219
                                          ;15220  ; ************************************************
                                          ;15221  ; * Patch no. 095, PCS 06E4 trapped to WCS 119E *
                                          ;15222  ; ************************************************
                                          ;15223
                                          ;15224          ;101-----------------------------;DEST ADDR = BYTE 1, FILL
                                          ;15225          LAB_R[R1],                        ;NECESSARY FOR FPD INTERFACE
                                          ;15226          Q_Q-K[.1],                        ;ROOM FOR 1 BYTE?
                                          ;15227          C[K.UBCC,
                                          ;15228          STATE_STATE.OR.K[.1],
U 06E5, 0019,2000,05C0,FA08,1414,25EE     ;15229          J/MOVCFILLWR
                                          ;15230
                                          ;15231          ;110-----------------------------;DEST ADDR = BYTE 2, FILL
                                          ;15232          LAB_R[R1],                        ;NECESSARY FOR FPD INTERFACE
                                          ;15233          Q_D+K[.2],                        ;UPDATE COUNTER FOR A WORD MOVE
                                          ;15234          C[K.UBCC,                         ;WILL THAT FIT?
                                          ;15235          STATE_STATE.OR.K[.2],
                                          ;15236          C31?,
U 06E6, 0019,0314,09C0,FA08,1414,25EC     ;15237          J/MOVCFILLMORE
                                          ;15238
                                          ;15239          ;111-----------------------------;DEST ADDR = BYTE 3, FILL
                                          ;15240          LAB_R[R1],                        ;NECESSARY FOR FPD INTERFACE
                                          ;15241          Q_Q-K[.1],
                                          ;15242          C[K.UBCC,
                                          ;15243          STATE_STATE.OR.K[.1],
U 06E7, 0019,2000,05C0,FA08,1414,25EE     ;15244          J/MOVCFILLWR
```

```
                                         =0*     ;--------------------------------;ALU <C>
                                         MOVCFWC:
                                                 D_Q-K[.2],                       ;NOT ENOUGH ROOM FOR A LWD
                                                 C[K.UBCC,
                                                 D(1)?,                           ;ROOM FOR 1 BYTE OR 1 WORD?
U 05C8, 0819,2C00,0980,F800,0010,05D9            J/MOVCFB

                                                 ;1*----------------------------;
                                         MOVCFL:
                                                 D[LONG]_CACHE,
                                                 Q_D,                             ;DECREMENTED COUNTER
                                                 R[R1]_LA+K[.4],                  ;UPDATE SRC ADDR FOR NEXT REFERENCE
                                                 BEN/A[U1-0,                      ;CHECK ON BYTE OFFSET OF SRC ADDR
U 05CA, 0018,1514,11E0,4288,0000,06CB            J/MOVCFLUA

                                         =*01    ;--------------------------------;D<1> VIA BEN/MUL, SC IS ZERO
                                         MOVCFB:
                                                 R[R1]_LA+K[.1],
U 05D9, 0018,0014,0580,FA88,0000,0A06            J/MOVCRDBYTE

                                                 ;*11----------------------------;
                                         MOVCFWD:
U 05DB, 0018,0014,0980,FA88,0000,0A04            R[R1]_LA+K[.2]                   ;UPDATE SRC ADDR TO REFLECT THIS READ

                                                 ;--------------------------------;
                                                 D[WORD]_CACHE,
                                                 Q_D,                             ;COPY DECRMENTED COUNTER
                                                 STATE_STATE.OR.K[.2],
U 0A04, 0000,403C,09E0,4000,1404,26CF            J/MOVC'  'ITE

                                                 ;--------------------------------;
                                         MOVCRDBYTE:
                                                 D[BYTE]_CACHE,
                                                 Q_Q-K[.1],
                                                 C[K.UBCC,
                                                 STATE_STATE.OR.K[.1],
U 0A06, 0019,A000,05C0,4000,1414,26CF            J/MOVCFWRITE

                                         ; ************************************************
                                         ; * Patch no. 033, PCS 0A06 trapped to WCS 1165 *
                                         ; ************************************************

                                         =0*     ;-------------------- ---------;ALU <C>
                                         MOVCWD:                                  ;DEST ON WORD (10) BOUNDARY
                                                                                  ;CHECK FOR AT LEAST 2 BYTES TO MOVE
                                                 D_Q-K[.2],
                                                 C[K.UBCC,
                                                 D(1)?,                           ;D<1> VIA BEN/MUL
U 05DC, 0819,2C00,0980,F800,0010,05D9            J/MOVCFB

                                                 ;1*----------------------------;
                                                 D_Q-K[.2],
                                                 C[K.UBCC,
U 05DE, 0819,2000,0980,F800,0010,05DB            J/MOVCFWD
```

```
                                            ;15299  ;SC = 0
                                            ;15300  ;Q + ID[T2] = COUNT - 4
                                            ;15301  ;ALU Z SET ON CURRENT CONTENTS OF Q
                                            ;15302  ;LAB = CURRENT(UNALIGNED) SRC ADDR
                                            ;15303  ;R1 = NEXT SRC ADDR(ALSO UNALIGNED), I.E LAB + 4
                                            ;15304  ;D = CORRECT DATA FOR NEXT WRITE, I.E APPROPRIATELY ALIGNED
                                            ;15305  ; BECAUSE OF 1 READ/LONG THAT WAS NOT AT BYTE 0 OFFSET
                                            ;15306
                                            ;15307  =1011   ;-------------------------------;ALU <1:0>
                                            ;15308  MOVCFLUA:
                                            ;15309          VA_LA.ANDNOT.K[.3],             ;ALWAYS READ ALIGNED
                                            ;15310          SC_K[.3],
U 06CB, 0018,0024,0D80,F800,0284,6A08      ;15311          J/MOVCFLUA1
                                            ;15312
                                            ;15313          ;1111--------------------------;
                                            ;15314  MOVCFWRITE:
                                            ;15315          LA_RA[3],                       ;UPDATE LA ONLY
                                            ;15316          VA_LA,
                                            ;15317          INTRPT.STROBE,                  ;INTERRUPTS PENDING?
                                            ;15318          FE_SC,                          ;SAVE SC FOR A STATE OR SO
                                            ;15319          STATE1-0?,                      ;HOW MANY BYTES TO WRITE?
U 06CF, 0000,173C,0180,F898,4300,0728      ;15320          J/MOVCWRITE
                                            ;15321
                                            ;15322  =**00   ;-------------------------------;STATE <1:0>
                                            ;15323  MOVCWRITE:
                                            ;15324          CACHE_D[LONG],                  ;4 BYTES TO WRITE
                                            ;15325          R[R3]_LA+K[.4],
                                            ;15326          BEN/INTERRUPT,
U 0728, 0018,0E14,1180,3298,0000,06B6      ;15327          J/MOVCFLP
                                            ;15328
                                            ;15329          ;**01--------------------------;1 BYTE TO WRITE
                                            ;15330          CACHE_D[BYTE],
                                            ;15331          STATE_STATE.ANDNOT.K[.1],
                                            ;15332          ALU_K[.1],SC_ALU,               ;SC_1 FOR ADDR INCREMENTATION
                                            ;15333          BEN/INTERRUPT,
U 0729, 0018,8E38,0580,3000,1486,46F6      ;15334          J/MOVCBWUPDATE
                                            ;15335
                                            ;15336          ;**10--------------------------;2 BYTES TO WRITE
                                            ;15337          CACHE_D[WORD],
                                            ;15338          STATE_STATE.ANDNOT.K[.2],
                                            ;15339          ALU_K[.2],SC_ALU,               ;SC_2 FOR ADDR INCREMENTATION
                                            ;15340          BEN/INTERRUPT,
U 072A, 0018,4E38,0980,3000,1486,46F6      ;15341          J/MOVCBWUPDATE
                                            ;15342  =
                                                   ;-------------------------------;
                                        =110                                       ;INTERRUPTS?
                                        MOVCBWUPDATE:
                                                   R[R3]_LA+K[SC],
                                                   D_ALU,                          ;GET BYTE OFFSET FOR BEN/MUL
                                                   SC_FE,                          ;RESTORE SC
                                                   Z?,                             ;ANY LEFT?
U 06F6, 0818,0114,1D80,FA98,0081,06C6              J/MOVCFTST

                                                   ;111---------------------------;
                                                   R[R3]_LA+K[SC],                 ;THERE'S AN INTERRUPT PENDING
                                                   D_Q,                            ;RESTORE R3 AND PUT LOOPCOUNT IN D
U 06F7, 0C18,0014,1D80,FA98,0000,0733              J/MOVCPACKST                    ;GO PACK R0 AND HONOR INT.

                                        =0*        ;-------------------------------;ALU <C>
                                        MOVCFILLMORE:
                                                   Q_D+K[.2],                      ;Q_COUNT-4+2
                                                   STATE_STATE.OR.K[.2],
                                                   CLK.UBCC,
                                                   D(1)?,                          ;IS THERE 1 BYTE OR 1 LWD LEFT?
U 05EC, 0019,0C14,09C0,F800,1414,2735              J/MOVCFILLBYTE

                                                   ;1*----------------------------;
                                        MOVCFILLWR:
                                                   LA_RA[5],                       ;PRESERVE LB WITH R1 FOR FAULTS + INTERRUPTS
                                                   D_LA,
U 05FE, 0800,003C,0180,F8A8,0000,06CF              J/MOVCFWRITE

                                        =101       ;-------------------------------;D<1> VIA BEN/MUL
                                        MOVCFILLBYTE:                              ;ONLY 1 BYTE LEFT
                                                   Q_Q+K[.1],                      ;Q_COUNT-2+1
                                                   CLK.UBCC,
                                                   STATE_STATE-K[.1],              ;CLEAR STATE <1> + SET STATE <0>
U 0735, 0019,2014,05C0,F800,1414,A5EE              J/MOVCFILLWR

                                                   ;111---------------------------;
                                                   LA_RA[5],                       ;AT LEAST 1 WORD LEFT
                                                   D_LA,
U 0737, 0800,003C,0180,F8A8,0000,06CF              J/MOVCFWRITE                    ;GO MOVE 1 WORD
```

```
                                        ;15382  MOVCFLUA1:
                                        ;15383          VA_VA+4,                        ;PEPARE TO READ NEXT LWD ALIGNED
                                        ;15384          ALU_0+MASK+1,                   ;SC=3 SO CREATE FFFFFFF8,
                                        ;15385          RC[T0]_ALU.RIGHT,SI/ASHR,       ;SHIFT RIGHT TO GET FFFFFFFC (-4)
                                        ;15386          SC_K[.FFE0],                    ;NEED -32 TO GET CORRECT DATA FROM
U 0A08, 0043,0010,A080,F983,0084,661A   ;15387          J/MOVCUNLRD                     ;Q INTO D FOR NEXT WRITE
                                        ;15388
                                        ;15389  =0*     ;-------------------------------;ALU <C>
                                        ;15390  MOVCUNALMORE:                           ;NO MORE UNALIGNED SRC MOVES
                                        ;15391          VA_LA,                          ;RESTORE VA
                                        ;15392          SC_K[ZERO],                     ;+ SC
                                        ;15393          D_Q,                            ;+ D
U 0618, 0C00,003C,1980,F800,0284,65C8   ;15394          J/MOVCFWC                       ;+ JOIN MAIN FORWARD LOOP
                                        ;15395
                                        ;15396          ;1*-----------------------------;
                                        ;15397  MOVCUNLRD:
                                        ;15398          D[LONG]_CACHE,
                                        ;15399          Q_D,                            ;COPY PREVIOUS ALIGNED READ DATA
                                        ;15400          LC_RC[T0]&R1_LA+K[.4],          ;UPDATE FOR NEXT READ
                                        ;15401          BEN/ROR,                        ;CHECK LA<1:0>
U 061A, 0018,0214,11E0,4380,0000,0742   ;15402          J/MOVCUNSHF
                                        ;15403
                                        ;15404          ;-------------------------------;
                                        ;15405  =010                                    ;LA<1:0>
                                        ;15406  MOVCUNSHF:
                                        ;15407  =011     ;------------------------------;LA = 01
                                        ;15408          LA_RA[3],                       ;PRESERVE R1 IN LB
                                        ;15409          VA_LA,
                                        ;15410          INTRPT.STROBE,
                                        ;15411          D_DAL.SC, Q_D,
                                        ;15412          SC_K[.18],                      ;SC = 24 DEC
U 0743, 0D00,003C,7DE0,F898,4284,6A09   ;15413          J/MOVCUNWRITE
                                        ;15414
                                        ;15415          ;110----------------------------;LA <1:0> = 10
                                        ;15416          LA_RA[3],
                                        ;15417          VA_LA,
                                        ;15418          INTRPT.STROBE,
                                        ;15419          D_DAL.SC, Q_D,
                                        ;15420          SC_K[.10],
U 0746, 0D00,003C,65E0,F898,4284,6A09   ;15421          J/MOVCUNWRITE
                                        ;15422
                                        ;15423          ;111--------------    ---------;LA <1:0> = 11
                                        ;15424          LA_RA[3],
                                        ;15425          VA_LA,
                                        ;15426          INTRPT.STROBE,
                                        ;15427          D_DAL.SC, Q_D,
                                        ;15428          SC_K[.8],
U 0747, 0D00,003C,01E0,F898,4284,6A09   ;15429          J/MOVCUNWRITE
                                        ;15430  =
                                        ;15431          ;-------------------------------;
                                        ;15432  MOVCUNWRITE:
                                        ;15433          CACHE_D[LONG],                  ;WRITE A LONGWORD ALIGNED
                                        ;15434          R[R3]_LA+K[.4],                 ;INCR DEST ADDR .
                                        ;15435          BEN/INTERRUPT,                  ;NOW BYTE 0 OF ''NEXT'' LWD
U 0A09, 0018,0E14,1180,3298,0000,0726   ;15436          J/MOVCUNINT
```

```
                                       ;15437  =110     ;------------------------------;INTERRUPTS?
                                       ;15438  MOVCUNINT:                               ;NO INTERRUPTS PENDING
                                       ;15439           D_Q,                            ;D_ MOST RECENT LWD READ
                                       ;15440           Q_ID[T2],                       ;LOAD COUNTER
                                       ;15441           LAB_R[R1],                      ;LATCH SRC ADDR
                                       ;15442           VA_[A.AND.LC,                   ;MASK OUT LOW BITS
                                       ;15443           Z?,                             ;MORE TO DO?
U 0726, 0C10,0134,C9F0,2E08,0200,05BC  ;15444           J/MOVCUNMORE
                                       ;15445
                                       ;15446           ;111---------------------------;
                                       ;15447           Q_ID[T2],                       ;GET UN-INCREMENTED COUNTER
U 0727, 0000,003C,C9F0,2C00,0000,06B7  ;15448           J/MOVCINTF
                                       ;15449
                                       ;15450  =0       ;------------------------------;ALU <Z>
                                       ;15451  MOVCUNMORE:
                                       ;15452           RC[T2]_Q-K[.4],                 ;ANOTHER LWD LEFT TO DO?
                                       ;15453           D_Q-K[.4],
                                       ;15454           Q_D,                            ;MOST RECENT LWD READ
                                       ;15455           C[K.UBCC,
                                       ;15456           VA_VA+4,
U 05BC, 0819,2000,11E0,F993,0010,0A0A  ;15457           J/MOVCUNOTHER
                                       ;15458
                                       ;15459           ;1-----------------------------;COUNT = 0
                                       ;15460           D_NOT.R[R2], Q_NOT.R[R2],       ;GET -(FILL COUNT) IN D AND Q LOW
                                       ;15461           STATE_STATE.ANDNOT.K[.30],
                                       ;15462           BEN/STATE7-4,
U 05BD, 0800,1628,79C0,F890,1404,47A3  ;15463           J/MOVCMAYBEFILL
                                       ;15464
                                       ;15465           ;------------------------------;
                                       ;15466  MOVCUNOTHER:
                                       ;15467           ID[T2]_D,                       ;SAVE DECREMNTED COUNTER
                                       ;15468           Q_D-LC,                         ;RESTORE COUNTER(LC=-4)
                                       ;15469           D_Q,
                                       ;15470           C31?,
U 0A0A, 0C11,0300,C9C0,3C00,0000,0618  ;15471           J/MOVCUNALMORE
```

```
                                        .TOC     ''        Character string       : MOVC3/5 BACKWARDS MOVE''

                                        =00      ;--------------------------------;MOVCRBUMP CALL CONSTRAINT
                                        MOVCBCKWDSMAYBE:
                                                 R[R5]_LC,                         ;COUNTER
                                                 Q_LC,
                                                 C[K.UBCC,                         ;MAY BE 0 SO CHECK IT
                                                 C31?,                             ;DETERMINE IF FWD OR BCKWRD
U 0770, 0010,0339,01C0,FAA8,0010,05C4            CALL,J/MOVCRBUMP

                                                 ;--------------------------------;
                                        =10                                        ;ALU <Z>
                                        MOVCRLP:
                                                 LAB_R1&RC[T2]_Q-K[.4],            ;MORE TO DO
                                                 D_Q-K[.4],                        ;DECREMENT COUNTER
                                                 C[K.UBCC,
                                                 INTRPT.STROBE,
                                                 BEN/SC,                           ;SC > 0?
U 0772, 0819,3400,1180,FB10,4010,0765            J/MOVCBCKSRC

                                                 ;11------------------------------;ALL DONE
                                        EXITR:   Q_R[R5],                          ;ALL DONE WITH SRC MOVE
U 0773, 0000,003C,01C0,FA28,0000,06C4            J/MOVCREADJUST

                                        =101     ;--------------------------------;SC GT 0
                                        MOVCBCKSRC:
                                                 R[R1]&VA_LA-K[.4],                ;SC=0 - MOVE A LWD
                                                 SC_K[.4],                         ;ASSUME IT'S A LWD
                                                 C31?,                             ;TEST IF 4 OR MORE BYTES LEFT
U 0765, 0018,0300,1180,FA88,0284,661C            J/MOVCBCKSRC2

                                                 ;111-----------------------------;
                                                 R[R1]&VA_LA-K[.1],                ;DECREMENT SRC ADDR FOR 1 BYTE'S WORTH
                                                 SC_K[.1],                         ;SET BYTE FLAG
U 0767, 0018,0000,0580,FA88,0284,6A0C            J/MOVCRDBCK1

                                        =0*      ;--------------------------------;ALU <C>
                                        MOVCBCKSRC2:
                                                 R[R1]&VA_LA-K[.1],                ;A BYTE
                                                 SC_K[.1],                         ;SET BYTE FLAG
U 061C, 0018,0000,0580,FA88,0284,6A0C            J/MOVCRDBCK1

                                                 ;1*------------------------------;A LWD
                                                 D[LONG]_CACHE,                    ;READ A LWD
                                                 Q_D,                              ;DUPLICATE COUNTER
                                                 LA_RA[R3],                        ;LATCH DEST ADDR
                                                 STATE_STATE.ANDNOT.K[.1],                  ;NOTE IT'S NOT A BYTE
                                                 BEN/INTERRUPT,
U 061E, 0000,0E3C,05E0,4098,1404,47B6            J/MOVCBCKWRITE
```

```
                                        ;15521  =110     ;-------------------------------;INTERRUPT?
                                        ;15522  MOVCBCKWRITE:
                                        ;15523          R[R3]&VA_LA-K[SC],              ;DECREMENT DEST ADDR FOR BYTE OR LONG
                                        ;15524          SC_ALU,                         ;PREPARE LO BIT FLAG
                                        ;15525          BEN/STATE3-0,                   ;BRANCH ON LWD OR BYTE
U 07B6, 0018,1700,1D80,FA98,0282,05FC   ;15526          J/MOVCBCKWR1
                                        ;15527
                                        ;15528          ;111----------------------------;INTERRUPT PENDING
                                        ;15529          LC_RC[T2]&R1_LB,
U 07B7, 000C,0038,0180,FB90,0000,0A11   ;15530          J/MOVCINTR
                                        ;15531
                                        ;15532  =***0    ;-------------------------------;STATE <3:1> NEVER SET
                                        ;15533  MOVCBCKWR1:
                                        ;15534          CACHE_D[LONG],
                                        ;15535          SC_SC.ANDNOT.K[.FFFC],          ;PRESERVE BITS <1:0>
                                        ;15536          Z?,                             ;MORE TO DO?
U 05FC, 0000,013C,F180,3000,0084,4772   ;15537          J/MOVCRLP
                                        ;15538
                                        ;15539          ;***1---------------------------;BYTE WRITE
                                        ;15540          CACHE_D[BYTE],
                                        ;15541          SC_SC.ANDNOT.K[.FFFC],
                                        ;15542          Z?,
U 05FD, 0000,813C,F180,3000,0084,4772   ;15543          J/MOVCRLP
                                        ;15544
                                        ;15545          ;-------------------------------;
                                        ;15546  MOVCRDBCK1:
                                        ;15547          D[BYTE]_CACHE,                  ;READ 1 BYTE
                                        ;15548          LA_RA[R3],
                                        ;15549          Q_Q-K[.1],                      ;DECREMENT COUNTER FOR 1 BYTE
                                        ;15550          C[K.UBCC,
                                        ;15551          STATE_STATE.OR.K[.1],           ;NOTE IT'S A BYTE OPERATION
                                        ;15552          BEN/INTERRUPT,
U 0A0C, 0019,AE00,05C0,4098,1414,27B6   ;15553          J/MOVCBCKWRITE
                                        ;15554
                                        ;15555  ; *********************************************
                                        ;15556  ; * Patch no. 032, PCS 0A0C trapped to WCS 1164 *
                                        ;15557  ; *********************************************
                                        ;       COME HERE WHEN MOVE LOOP EXHAUSTED - BEN ON STATE<6>
                                        ;       TO TELL WHETHER TO FILL OR NOT.
                                        ;       D AND Q HAVE -(FILL COUNT) IN BITS 15:0, AND THE COMPLEMENT
                                        ;       OF 2 FILL CHARACTERS IN 31:16.

                                        =*011   ;--------------------------------;STATE <6>
                                        MOVCMAYBEFILL:

                                                ;--------------------------------;
                                        ;011                                     ;NO FILL NECESSARY
                                        ;       PC&VA_PC, FLUSH.IB,              ;RESET IB (ONLY NECECESSARY IF WE
                                        ;               CLR.FPD, J/MOVCEXIT      ; WERE RESTARTED) AND GO CLEAR REGS

                                                ;--------------------------------;
                                                R[R0]_Q.AND.K[.FFFF],            ;USE THIS CODE UNTIL YOU
U 07A3, 0019,2034,C180,FA80,0000,09EE          J/R245ZERO                       ;UNDERSTAND HOW THE IB WORKS.

                                                ;111-----------------------------;NEED TO FILL
U 07A7, 0B1F,0000,6580,FAF8,0084,6A0D          R[R15]_0-Q, D_D.SWAP, SC_K[.10] ;NEED TO FILL. R2 HAD -(FILL CT)
                                                                                 ;IN <15:0>; NEGATE IT AND SET UP
                                                                                 ;TO REPLICATE FILLS FROM R2<31:16>

                                                ;--------------------------------;
                                                Q_R[R15].AND.K[.FFFF],           ;CLEAR HI-ORDER CRUD FROM COUNT,
U 0A0D, 0D18,0034,C1C0,FA78,0010,0A0E          C[K.UBCC, D_DAL.SC               ;SET Z ON IT, PUT 4 FILLS IN D

                                                ;--------------------------------;
                                                R[R5]_NOT.D,                     ;STORE FILLS IN R5, CLEAR BIT 6
U 0A0E, 0001,0028,7580,FAA8,1404,AA10          STATE_STATE-K[.20]               ;(NEED TO FILL), SET BIT 5 (FILLING)

                                                ;--------------------------------;
U 0A10, 0018,0038,C180,FA90,0000,06B6          R[R2]_K[.FFFF], J/MOVCFLP        ;SET UP R2 FOR NEXT EXIT. GO FILL.
                                                                                 ;(SC .NE. 0 TO INDICATE FILLS)


                                                ;--------------------------------;

                                        ;MOVCEXIT:                               ;THIS IS THE END.......
                                        ;       R[R0]_Q.AND.K[.FFFF],            ;SET R0 TO MAX(SRCLEN-DSTLEN,0)
                                        ;               PC_PC+1,LOAD.IB          ;START RELOADING INSTRUCTION BUFFER
                                        ;
                                        ;       R[R5]_0, D_0, Q_0,               ;ZERO R5 AND PREPARE TO ZERO
                                        ;               J/MOVGETOUT              ;R2 AND R4 AND EXIT VIA IRD.
                                        .TOC    "       Character string      : MOVC3/5, MOVTC, MOVTUC FPD"

                                        ;THE FAULT VECTOR FOR MOVC3/5 + MOVTC/TUC IS FORCED TO BE F80.


                                                ;--------------------------------;
                                        0F81:                                    ;WRITE FAULT ENTRY POINT
U 0F81, 0000,003C,0180,FA98,0000,0F82           R[R3]_LA                         ;SAVE START-OF-ITERATION DSTADR

                                                ;--------------------------------;
                                        0F82:                                    ;READ FAULT ENTRY POINT
                                        MOVC.RDFAULT:
U 0F82, 000C,0038,0183,FB90,0000,0A11           LC_RC[T2]&R1_LB, SD_NOT.SD       ;SAVE START-OF-ITERATION SRCADR
                                                                                 ;SET FAULT FLAG (IN SD), GET COUNT-4

                                                ;--------------------------------;
                                        MOVCINTR:                                ;BACKWARDS MOVE INTERRUPT ENTRY
                                                D_O+LC+1,                        ;CURRENT COUNT MINUS 4 IN RC[T2]
U 0A11, 0813,0910,0180,F800,0000,0732           IR1?                             ; (MINUS 1 IF MOVTC OR MOVTUC)

                                        =10     ;--------------------------------;IR <1>
U 0732, 0819,0014,0D80,F800,0000,0733           D_D+K[.3]                        ;GET TRUE COUNT TO STORE IN R0

                                                ;11------------------------------;
                                        MOVCPACKST:                              ;COMMON ENTRY FOR REGISTER PACKING
                                                D_D.SWAP, Q_PC, SC_K[ZERO],      ;PUT LOOP COUNT IN D HIGH,
U 0733, 0B14,0038,19C2,F800,0094,6624           SS_SD, CLK.UBCC                  ;SS = FAULT FLG, CLR EALU.N & SC

                                                ;--------------------------------;
                                        =0*                                      ;CONSTRAINT BLOCK FOR CALL
                                                R[R0]_D,                         ;SAVE LOOPCOUNT IN R0<31:16>
                                                STATE_STATE.ANDNOT.K[.3],              ;CLEAR STATE <1:0> SO IT CAN BE
                                                                                 ;OR'D WITH PC DELTA VIA THE
                                                                                 ;BMUX USING THE PACK.FLOAT LEG.
U 0624, 0001,003D,0D80,FA80,1404,4EB8           CALL,J/BAKUP.PC                  ;BAKUP.PC RETURNS PC DELTA IN D

                                                ;1*------------------------------;
                                                EALU_STATE,                      ;FETCH, EFFECTIVELY, STATE <7:2>
                                                R[R0]_D.OR.PACK.FP,              ;COMBINE PC DELTA + STATE INTO
U 0626, 0009,5230,0180,FA80,1400,00F4           DT/WORD, SS?                     ;R0<15:0> AND CHK FAULT OR INT

                                        =*1*0   ;--------------------------------;SIGN SRC(SS)
U 00F4, 0000,0E3C,0180,F800,0000,0F8D           BEN/INTERRUPT, J/INTIO           ;INTERRUPT - SEE WHICH KIND

                                        ; ***********************************************
                                        ; * Patch no. 030, PCS 00F4 trapped to WCS 1162 *
                                        ; ***********************************************

                                                ;*1*1----------------------------;
U 00F5, 0000,003C,0180,F800,0000,0EB8           J/FPD.RTN                        ;FAULT - TAKE THE EXCEPTION.
```

```
                                          ;MOVC/MOVTC/MOVTUC RESTART CODE - COME HERE FROM IRD IF FPD SET.

                                                  ;-------------------------------;
                                          48:
                                          MOVCRESTART:
                                                  LA_RA[R0],                        ;R0 CONTAINS STATE + PC DELTA
                                                  D_[A,                             ; + LOOPCOUNT BYTE-SWAPPED
U 0048, 0800,003C,0180,F880,1408,6A12             STATE_AMX.EXP                     ;RESTORE STATE

                                                  ;-------------------------------;
                                                  PC&VA_D.OXT[BYTE]+PC, D_D.SWAP,   ;UPDATE PC AND UNSWAP LOOP COUNT
                                                  STATE_STATE.ANDNOT.K[.F],         ;STATE BITS <3:0> NOT OF INTEREST
U 0A12, 0B17,8014,6180,F801,1684,4A14             SC_STATE.ANDNOT.K[.F]             ;SC GETS FLAG BITS FOR FWD MOVE

                                                  ;-------------------------------;
U 0A14, 0858,0038,91E0,F800,0000,0A15             Q_D, D_K[.1F00].RIGHT             ;SAVE COUNT, GET FAULT VECTOR

                                                  ;-------------------------------;
                                                  ID[FPDA]_D, D_Q.OXT[WORD],        ;ISOLATE LOOP CT IN WD, SET VECTOR
                                                  SS_0&SD_0, CLK.UBCC,              ;CLEAR FAULT FLAG AND SET Z ON CT
U 0A15, 0803,763C,B587,3C00,0010,06D6             STATE7-4?                         ;TEST MOV VS MOVT AND DIRECTION

                                          =0110   ;-------------------------------;STATE7-4
                                                  SC_SC.ANDNOT.K[.50], Q_D,         ;FWD MOVC - SET SC TO FILL FLAG
U 06D6, 0000,003C,35E0,F800,0084,46B6             J/MOVCFLP

                                                  ;0111---------------------------;
                                                  SC_R[R3].AND.K[.3], Q_D,          ;BKWD MOVC - SC=DEST ADDR<1:0>
U 06D7, 0018,0034,0DE0,FA18,0082,0772             J/MOVCRLP

                                                  ;1110---------------------------;
                                                  D_R[R2], ID[T0]_D, Q_0,           ;FWD MOVTC/TUC - GET FILL/ESC,
U 06DE, 0800,003C,C1F8,3E10,0000,032E             J/MOVTCFWD                        ;SAVE COUNT AND RE-ENTER LOOP

                                                  ;1111---------------------------;
                                                  D_R[R2], ID[T0]_D, Q_0,           ;BKWD MOVTC - GET FILL,
U 06DF, 0800,003C,C1F8,3E10,0000,032A             J/MOVTCBKWD                       ;SAVE COUNT AND RE-ENTER LOOP
```

```
.TOC    "       Character string        : SKPC, LOCC"

;SKPC TIL UNEQUAL; LOCC TIL EQUAL

;ALGORITHM:
;THE SOURCE IS COMPARED WITH THE MASK CHARACTER TIL FOUND/NOT FOUND,
;DEPENDING ON THE OP-CODE. THIS SEARCH IS CONDUCTED BY BYTES TIL A
;LONGWORD BOUNDARY IS REACHED, AT WHICH TIME IT IS CONTINUED AS
;LONGWORDS TIL < 4 BYTES REMAIN TO BE SEARCHED, WHEN IT REVERTS
;TO BYTE-WISE SEARCH AGAIN.

;INPUTS:
;Q       CHARACTER FOR THE COMPARISON(1ST OPERAND)
;D       NUMBER OF BYTES TO COMPARE(2ND OPERAND)

;REGISTER USAGE:
;R0      BYTE 1-0 = SRC LEN
;        BYTE 2 = PC DELTA FOR FPD
;        BYTE 3 = COMP CHAR FOR FPD
;R1      SRC ADDR
;Q       LENGTH
;RC 2    COMPARE CHAR

;OUTPUTS:
;R0      NUMBER OF BYTES REMAINING IF BYTE LOCATED OR 0 IF NOT LOCATED
;R1      ADDRESS OF BYTE LOCATED + 1 OR END OF STRING + 1

;LABELS OF INTEREST:
;SKPRES1         RESUME EXECUTION AFTER RECOVERING FROM AN INTERRUPT/EXCEPTION
;SKPBYTES        READ + COMPARE BY BYTES LOOP
;SKPALIGNED      START OF LWD COMPARES. MAKE A LWD OF COMPARE CHAR
;SKPLONGLOOP     READ + COMPARE BY LWDS LOOP
```

```
                                        ;15718  488:    ;-------------------------------;
                                        ;15719          RC[T2]_Q.AND.K[.FF],            ;1ST ARG IS COMPARE BYTE
U 0488, 0019,2035,4980,F990,0000,047E   ;15720          CALL,J7ASPC                     ;GET 3RD ARG
                                        ;15721
                                        ;15722  4E8:    ;-------------------------------;
U 04E8, 0019,2034,C180,FA80,0000,02B8   ;15723          R[R0]_Q.AND.K[.FFFF]            ;2ND ARG IS LENGTH
                                        ;15724
                                        ;15725  =0****00 ;------------------------------;RETURN40 + RETURN2
                                        ;15726          R[R1]_D,                        ;ARG 3 IS ADDR
                                        ;15727          Q_ID[PSL],                      ;PREPARE TO CLEAR PSL CC
U 02B8, 0001,003D,3DF0,2E88,0000,09E4   ;15728          CALL,J/CLRPSLCC                 ;
                                        ;15729
                                        ;15730  =1****00 ;------------------------------;RETURN2 NEEDED
                                        ;15731  SKPRES1:                                ;FPD RESTART LOCATION
                                        ;15732          CALL,J/SETFPD,                  ;SET FPD BIT
                                        ;15733          D_R[R0].AND.K[.FFFF],           ;GUARANTEE IT'S A WORD FOR RESTART ALSO
U 02F8, 0818,0035,C180,FA00,0010,0E16   ;15734          C[K.UBCC                        ;CHECK ON SRC LENGTH
                                        ;15735
                                        ;15736          ;-------------------------------;
U 02F9, 0000,003C,0180,F800,0000,0A26   ;15737          J/SKPFPD                        ;SKPC/LOCC FPD ADDR
                                        ;15738
                                        ;15739          ;-------------------------------;
U 02FA, 0000,003C,0180,F800,0000,0A26   ;15740          J/SKPFPD                        ;SKPC/LOCC FPD ADDR
                                        ;15741
                                        ;15742          ;-------------------------------;
                                        ;15743          Z?,                             ;BRANCH ON LENGTH > 0.
                                        ;15744          Q_D,                            ;COPY LENGTH
U 02FB, 0000,013C,01E0,FA08,0200,060C   ;15745          VA_R[R1]                        ;LOAD ADDR OF 1ST BYTE
```

```
B  1  I-stream decode forks : A-FORK for VAX Instructions
C  1  I-stream decode forks : A-FORK for VAX Instructions
D  1  I-stream decode forks : B-FORK for VAX Instructions
E  1  I-stream decode forks : B-FORK for VAX Instructions
F  1  I-stream decode forks : B-FORK for VAX Instructions
G  1  I-stream decode forks : B-FORK for VAX Instructions
H  1  I-stream decode forks : B-FORK for VAX Instructions
I  1  I-stream decode forks : B-FORK for VAX Instructions
J  1  I-stream decode forks : B-FORK for VAX Instructions
K  1  I-stream decode forks : B-FORK for VAX Instructions
L  1  I-stream decode forks : B-FORK for VAX Instructions
M  1  I-stream decode forks : B-FORK for VAX Instructions
N  1  I-stream decode forks : B-FORK for VAX Instructions
B  2  I-stream decode forks : B-FORK for VAX Instructions
C  2  I-stream decode forks : B-FORK for VAX Instructions
D  2  I-stream decode forks : B-FORK for VAX Instructions
E  2  I-stream decode forks : B-FORK for VAX Instructions
F  2  I-stream decode forks : B-FORK for VAX Instructions
G  2  I-stream decode forks : B-FORK for VAX Instructions
H  2  I-stream decode forks : B-FORK for VAX Instructions
I  2  I-stream decode forks : C-FORK for VAX Instructions
J  2  I-stream decode forks : C-FORK for VAX Instructions
K  2  I-stream decode forks : C-FORK for VAX Instructions
L  2  I-stream decode forks : C-FORK for VAX Instructions
M  2  I-stream decode forks : C-FORK for VAX Instructions
N  2  I-stream decode forks : C-FORK for VAX Instructions
B  3  I-stream decode forks : C-FORK for VAX Instructions
C  3  I-stream decode forks : C-FORK for VAX Instructions
D  3  I-stream decode forks : C-FORK for VAX Instructions
E  3  I-stream decode forks : C-FORK for VAX Instructions
F  3  I-stream decode forks : C-FORK for VAX Instructions
G  3  I-stream decode forks : C-FORK for VAX Instructions
H  3  I-stream decode forks : C-FORK for VAX Instructions
I  3  I-stream decode forks : C-FORK for VAX Instructions
J  3  I-stream decode forks : C-FORK for VAX Instructions
K  3  I-stream decode forks : C-FORK Specifier Evaluation Subrouti
L  3  I-stream decode forks : C-FORK Specifier Evaluation Subrouti
M  3  I-stream decode forks : C-FORK Specifier Evaluation Subrouti
N  3  I-stream decode forks : C-FORK Specifier Evaluation Subrouti
B  4  I-stream decode forks : C-FORK Specifier Evaluation Subrouti
C  4  I-stream decode forks : C-FORK Specifier Evaluation Subrouti
D  4  ARITH.MIC
E  4  Integer arithmetic    : Multiplication subroutine
F  4  Integer arithmetic    : Multiplication subroutine
G  4  Integer arithmetic    : Divide subroutine
H  4  Integer arithmetic    : MULB2, MULB3, MULW2, MULW3, MULL2, M
I  4  Integer arithmetic    : MULB2, MULB3, MULW2, MULW3, MULL2, M
J  4  Integer arithmetic    : MULB2, MULB3, MULW2, MULW3, MULL2, M
K  4  Integer arithmetic    : MULB2, MULB3, MULW2, MULW3, MULL2, M
L  4  Integer arithmetic    : EMUL
M  4  Integer arithmetic    : EMUL
N  4  Integer arithmetic    : EMUL
B  5  Integer arithmetic    : DIVB2, DIVB3, DIVW2, DIVW3, DIVL2, D
C  5  Integer arithmetic    : DIVB2, DIVB3, DIVW2, DIVW3, DIVL2, D
D  5  Integer arithmetic    : DIVB2, DIVB3, DIVW2, DIVW3, DIVL2, D
E  5  Integer arithmetic    : DIVB2, DIVB3, DIVW2, DIVW3, DIVL2, D
F  5  Integer arithmetic    : EDIV
G  5  Integer arithmetic    : EDIV
H  5  Integer arithmetic    : EDIV
I  5  INDEX.MIC

J  5  Index instruction     : INDEX
K  5  Index instruction     : INDEX
L  5  Index instruction     : INDEX
M  5  FLOAT.MIC
N  5  F & D floating point  : CMPF
B  6  F & D floating point  : CMPF
C  6  F & D floating point  : ADDF, SUBF
D  6  F & D floating point  : ADDF, SUBF
E  6  F & D floating point  : ADDF, SUBF
F  6  F & D floating point  : ADDF, SUBF
G  6  F & D floating point  : ADDF, SUBF
H  6  F & D floating point  : ADDF/SUBF ROUTINE
I  6  F & D floating point  : ADDF/SUBF ROUTINE
J  6  F & D floating point  : ADDF/SUBF ROUTINE
K  6  F & D floating point  : ADDF/SUBF ROUTINE
L  6  F & D floating point  : MULF
M  6  F & D floating point  : MULF
N  6  F & D floating point  : MULF
B  7  F & D floating point  : DIVF
C  7  F & D floating point  : DIVF
D  7  F & D floating point  : DIVF
E  7  F & D floating point  : CMPD
F  7  F & D floating point  : CMPD
G  7  F & D floating point  : UNPACK DOUBLE OPERANDS
H  7  F & D floating point  : UNPACK DOUBLE OPERANDS
I  7  F & D floating point  : UNPACK DOUBLE OPERANDS
J  7  F & D floating point  : UNPACK DOUBLE OPERANDS
K  7  F & D floating point  : PACK DOUBLE RESULT
L  7  F & D floating point  : PACK DOUBLE RESULT
M  7  F & D floating point  : ADDD, SUBD
N  7  F & D floating point  : ADDD, SUBD
B  8  F & D floating point  : ADDD, SUBD
C  8  F & D floating point  : ADDD, SUBD
D  8  F & D floating point  : ADDD, SUBD
E  8  F & D floating point  : ADDD, SUBD
F  8  F & D floating point  : ADDD, SUBD
G  8  F & D floating point  : ADDD, SUBD
H  8  F & D floating point  : ADDD, SUBD
I  8  F & D floating point  : ADDD, SUBD
J  8  F & D floating point  : ADDD, SUBD
K  8  F & D floating point  : MULD
L  8  F & D floating point  : DIVD
M  8  F & D floating point  : DIVD
N  8  F & D floating point  : DIVD
B  9  F & D floating point  : DIVD
C  9  F & D floating point  : DIVD
D  9  F & D floating point  : UNPACK ONE DOUBLE OPERAND
E  9  F & D floating point  : CVTBF, CVTWF, CVTLF
F  9  F & D floating point  : CVTBF, CVTWF, CVTLF
G  9  F & D floating point  : CVTBD, CVTWD, CVTLD
H  9  F & D floating point  : CVTFD, CVTDF
I  9  F & D floating point  : CVTFD, CVTDF
J  9  F & D floating point  : CVTFB, CVTFW, CVTFL, CVTRFL
K  9  F & D floating point  : CVTDB, CVTDW, CVTDL, CVTRDL
L  9  F & D floating point  : CVTDB, CVTDW, CVTDL, CVTRDL
M  9  F & D floating point  : CONVERT FLOATING TO INTEGER
N  9  F & D floating point  : CONVERT FLOATING TO INTEGER
B 10  F & D floating point  : CONVERT FLOATING TO INTEGER
C 10  F & D floating point  : CONVERT FLOATING TO INTEGER
D 10  F & D floating point  : ACBF
```

```
E 10  F & D floating point  : ACBF
F 10  F & D floating point  : ACBF
G 10  F & D floating point  : ACBF
H 10  F & D floating point  : ACBF
I 10  F & D floating point  : ACBD
J 10  F & D floating point  : ACBD
K 10  F & D floating point  : ACBD
L 10  F & D floating point  : MULD
M 10  F & D floating point  : MULD
N 10  F & D floating point  : MULD
B 11  F & D floating point  : MULD
C 11  F & D floating point  : MULD
D 11  F & D floating point  : EMODF
E 11  F & D floating point  : EMODF
F 11  F & D floating point  : EMODF
G 11  F & D floating point  : EMODF
H 11  F & D floating point  : EMODF
I 11  F & D floating point  : EMODF
J 11  F & D floating point  : EMODD
K 11  F & D floating point  : EMODD
L 11  F & D floating point  : EMODD
M 11  F & D floating point  : POLYF
N 11  F & D floating point  : POLYF
B 12  F & D floating point  : POLYF
C 12  F & D floating point  : POLYF
D 12  F & D floating point  : POLYF
E 12  F & D floating point  : POLYF
F 12  F & D floating point  : POLYF
G 12  F & D floating point  : POLYF
H 12  F & D floating point  : POLYD
I 12  F & D floating point  : POLYD
J 12  F & D floating point  : POLYD
K 12  F & D floating point  : POLYD
L 12  F & D floating point  : POLYD
M 12  F & D floating point  : POLYD
N 12  F & D floating point  : POLYD
B 13  F & D floating point  : POLYD
C 13  F & D floating point  : POLYD
D 13  F & D floating point  : POLYD
E 13  INIT2.MIC
F 13  Initialize microcode  :INITIALIZE MACHINE ROUTINE
G 13  Initialize microcode  :INITIALIZE MACHINE ROUTINE
H 13  Initialize microcode  :INITIALIZE MACHINE ROUTINE
I 13  Initialize microcode  :INITIALIZE MACHINE ROUTINE
J 13  Initialize microcode  :INITIALIZE MACHINE ROUTINE
K 13  ASPC.MIC
L 13  I-stream decode forks : Address Specifier Evaluation
M 13  I-stream decode forks : Address Specifier Evaluation
N 13  I-stream decode forks : Address Specifier Evaluation
B 14  I-stream decode forks : Address Specifier Evaluation
C 14  I-stream decode forks : Address Specifier Evaluation
D 14  FIELD.MIC
E 14  Field instructions    : FFS, FFC, CMPV, CMPZV, EXTV, EXTZV
F 14  Field instructions    : FFS, FFC, CMPV, CMPZV, EXTV, EXTZV
G 14  Field instructions    : FFS, FFC, CMPV, CMPZV, EXTV, EXTZV
H 14  Field instructions    : FFS, FFC, CMPV, CMPZV, EXTV, EXTZV
I 14  Field instructions    : FFS, FFC, CMPV, CMPZV, EXTV, EXTZV
J 14  Field instructions    : INSV
K 14  Field instructions    : INSV
L 14  Field instructions    : INSV
M 14  Field instructions    : INSV
N 14  Field instructions    : INSV
B 15  Field instructions    : INSV
C 15  CHAR.MI
D 15  Character string      : Utilities
E 15  Character string      : Utilities
F 15  Character string      : Utilities
G 15  Character string      : MOVC3, MOVC5
H 15  Character string      : MOVC3/5 INITIALIZATION
I 15  Character string      : MOVC3/5 INITIALIZATION
J 15  Character string      : MOVC3/5 INITIALIZATION
K 15  Character string      : MOVC3/5 INITIALIZATION
L 15  Character string      : MOVC3/5 MAIN LOOPS
M 15  Character string      : MOVC3/5 MAIN LOOPS
N 15  Character string      : MOVC3/5 MAIN LOOPS
B 16  Character string      : MOVC3/5 MAIN LOOPS
C 16  Character string      : MOVC3/5 MAIN LOOPS
D 16  Character string      : MOVC3/5 MAIN LOOPS
E 16  Character string      : MOVC3/5 MAIN LOOPS
F 16  Character string      : MOVC3/5 BACKWARDS MOVE
G 16  Character string      : MOVC3/5 BACKWARDS MOVE
H 16  Character string      : MOVC3/5 BACKWARDS MOVE
I 16  Character string      : MOVC3/5, MOVTC, MOVTUC FPD
J 16  Character string      : MOVC3/5, MOVTC, MOVTUC FPD
K 16  Character string      : SKPC, LOCC
L 16  Character string      : SKPC, LOCC
```